NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

METHODOLOGIES FOR THE HIGH RESOLUTION
MODELING OF
MINEFIELD DYNAMICS

by

Alan A. Anderson

September, 1991

Thesis Advisor: Samuel H. Parry

Approved for public release; distribution is unlimited

T256881

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL OR	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Including Security Classification)
METHODOLOGIES FOR THE HIGH RESOLUTION MODELING OF MINEFIELD DYNAMICS

12 PERSONAL AUTHOR(S)
Anderson, Alan Arthur

13 TYPE OF REPORT	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Year, Month, Day)	15. Page Count
Master's thesis		1991, September	183

16. SUPPLEMENTAL NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Simscript, Simgraphics, Minefields, Breaching, Simulation, Modeling, Navigation Error, Bypassing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Land mines are a continuing threat to the mobility required by the modern army. Efforts to develop solutions for the problems presented by mines are hampered by a lack of useful, realistic, high resolution models.

To assist in developing the needed modeling capabilities, several methodologies are proposed. Methodologies for modeling vehicle navigation error, mine encounters, plow displacement of mines, bypassing obstructions and the presence of overwatching direct fires are developed and explained. These methodologies are then implemented using SIMSCRIPT and SIMGRAPHICS into a minefield breaching model. The model will run in a graphics mode, allowing a visual validation of the model algorithms.

The problem of plow width versus breaching force casualty rates is examined as an example of the potential utility of the model.

20 DISTRIBUTION/AVAILABILTIY OF ABSTRACT	21a. REPORT SECURITY CLASSIFICATION	
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Samuel H. Parry	22b. TELEPHONE (Include Area Code) (408)646-2779	22c. OFFICE SYMBOL OR/Py

DD Form 1473, JUN 86 — Previous editions are obsolete. S/N 0102-LF-014-6603 — SECURITY CLASSIFICATION OF THIS PAGE Unclassified

Approved for public release; distribution is unlimited.

METHODOLOGIES FOR THE HIGH RESOLUTION MODELING OF MINEFIELD DYNAMICS

by

Alan A. Anderson
Major, United States Army
B.S., United States Military Academy

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

ABSTRACT

Land mines are a continuing threat to the mobility required by the modern army. Efforts to develop solutions for the problems presented by mines are hampered by a lack of useful, realistic, high resolution models.

To assist in developing the needed modeling capabilities, several methodologies are proposed. Methodologies for modeling vehicle navigation error, mine encounters, plow displacement of mines, bypassing obstructions and the presence of overwatching direct fires are developed and explained. These methodologies are then implemented using SIMSCRIPT and SIMGRAPHICS into a minefield breaching model. The model can be run in a graphics mode, allowing a visual validation of the model algorithms.

The problem of plow width versus breaching force casualty rates is examined as an example of the potential utility of the model.

A 444444
c.1

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION 1

II. PROBLEM DISCUSSION 5

A. MINES 5

1. Mine Designs. 6

2. Distinguishing Characteristics. 7

B. BREACHERS 9

C. DOCTRINE 13

1. Countermine Operations 13

2. Mine Operations 16

D. DISCUSSION 17

III. METHODOLOGY 20

A. VEHICLE NAVIGATION 21

B. MINE ENCOUNTERS 27

C. MINE DISPLACEMENT 29

D. OBSTRUCTION AVOIDANCE 31

E. OVERWATCHING FIRES 41

1. Direct Fires. 42

2. Indirect Fires. 43

IV. MODEL DESCRIPTION 44

A. ASSUMPTIONS AND MODEL LIMITATIONS 44

1. Uniform terrain 44

2. Constant vehicle speed 45

3. Obscuration 45

4. Breacher Types 46

5. Mine Types 46

6. Obstruction Types 46

B. PARAMETERS 47

1. Model Input Parameters 47

a. Mine parameters 48

b. Plow parameters 48

c. Vehicle parameters 49

d. Overwatch parameters 49

e. Taskforce parameters 50

f. Tables 51

g. Administrative parameters 51

2. Outputs 52

C. THE MODEL 53

1. Events 53

a. Encounter events 53

b. Fire events 56

V. EXAMPLE PROBLEM 58

A. SCENARIO 58
1. The Minefield 59
2. The Taskforce 59
3. Plowing 60
4. Red Overwatch 62
B. TEST PLAN 63
C. PROBLEM RESULTS 63

VI. CONCLUSIONS AND RECOMMENDATIONS 68
A. RESULTS 68
B. FUTURE ENHANCEMENTS 69

LIST OF REFERENCES 71

BIBLIOGRAPHY 72

APPENDIX A - SYSTEM REQUIREMENTS 73

APPENDIX B - RUNNING THE MODEL 74

APPENDIX C - SOURCE CODE LISTINGS 83

APPENDIX D - GRAPHIC ICONS 153

APPENDIX E - EXAMPLE SINGLE AVAILABLE OUTPUTS 160

APPENDIX F - EXAMPLE MULTIPLE RUN RESULTS 166

INITIAL DISTRIBUTION LIST 171

ACKNOWLEDGEMENT

I wish to thank Dr. John Farr of the Waterways Experimentation Station for his assistance in researching the history of the Engineer Model Improvement Program. Readers interested in obtaining a copy of the software code should contact Dr. Farr. His address is available in the initial distribution list. I also wish to thank Professor Sam Parry and LTC William Caldwell for their assistance in developing this thesis and for their guidance in its construction. Finally, I wish to thank my wife Susan for her support and understanding over the months of programming and writing that went into the model and this paper.

Alan A. Anderson

I. INTRODUCTION

> Currently, the Army's countermine capability is inadequate. Efforts to address this shortfall have been ongoing since the early 1980s and have resulted in some successes. Assessment of Operation Desert Storm requirements indicate that the mobility enhancement capability of the Army's detection and breaching assets needs improvement. [Ref. 1: p 72]

Mobility and maneuver are essential elements of the U.S. Army's Airland Battle Doctrine. Anything which restricts or limits our ability to maneuver is a potential hazard to the implementation of this doctrine. Land mines are such a hazard.

Events in the Middle East have recently highlighted deficiencies in our capability to effectively and rapidly respond to the problem presented by mines. In part this problem is due to the fact that the state of the military art in regards to countermine warfare has progressed very little since World War II. During this same period, advances in electronics and integrated circuits have enormously increased the capabilities and lethality of modern mines. Modern land mines are a largely unquantified impediment to the maneuverability required for successful implementation of the Airland Battle doctrine.

The deficiencies in the capabilities of the U.S. Army to conduct countermine warfare operations have long been recognized. Numerous ideas and programs have been suggested to alleviate the problem. Unfortunately, there is a corresponding lack of capability to effectively model mine and countermine scenarios and the proposed equipment. Since actual field experience is limited, and testing is both expensive and dangerous, modeling is a cost effective way to gain insights into the dual problems of dealing with, and of using, land mines. A brief background of the program currently

being under taken to remedy the modeling deficiencies is presented as a motivator for this thesis.

In 1979, the Review of Army Analysis [Ref. 2] found numerous deficiencies in the Army's computerized combat models. These deficiencies included poor documentation, poor response to study needs, inconsistent results, differing data assumptions, lack of interface structure, and limited (or no) functional area representation. As a result, a directive was published in April 1980, implementing an Army Model Improvement Program (AMIP). The tasks and responsibilities of AMIP are described in Army Regulation 5-11.

The Engineer Model Improvement Program (EMIP) is a part of the AMIP designed to ensure that the engineer functional area is properly represented in the Army's hierarchy of approved combat simulation models. The EMIP plan was published by the Engineer Studies Center [Ref. 3]. Major elements of the EMIP plan provide for changes to the Combined Arms and Support Task Force Evaluation Model (CASTFOREM), Vector-In-Command Model (VIC), the Force Evaluation Model (FORCEM), and for the development of an Engineer Functional Area Model (EFAM). Priority was placed on enhancements to VIC and the development of a VIC-based EFAM. This work was conducted by the US Army Corps of Engineer Laboratories and was completed in FY91.

Another goal of EMIP was to develop high resolution engineering specific simulation models. Deficiencies exist in analytical tools that can be used to perform studies of breaching operations of both linear (e.g., opposed river crossing operations) and area obstacles (e.g., obstacle complexes which might include tank ditches, concertina, minefields, etc.).

The CASTFOREM can be used for these types of studies, however, scenarios can require several man years worth of development time. The JANUS(A) interactive simulation model can be used for quick turn around studies. However, since JANUS(A) is an interactive simulation model, parametric studies are difficult if not impossible to perform.

Since the breaching of minefields is currently the highest priority within the engineering community, a major requirement exists for an analytical tool to conduct future material studies on such systems as the Combat Mobility Vehicle (CMV) and the Wide Area Mine (WAM), in support of the combat engineers. In addition, a tool is needed to address force structure (number of plows and rollers needed in a typical armor company) and tactical problems (location of breaching assets during various tactical formations) that presently need additional analysis.

The lack of high resolution tools to investigate minefield interactions is the stimulus for this thesis, which is intended to provide a baseline tool for use in investigating the dynamics of minefield breaching. Several basic methodologies for high resolution discrete modeling of minefield and vehicle interactions are proposed and then implemented in a SIMSCRIPT program. The resulting model is currently limited in scope, dealing solely with the more conventional mines and with plowing as the only breaching technique. However, the routines, functions and structure of the model provided facilitate the modeling of additional attributes of the minefield problem.

Interested readers are also directed to the thesis written by Captain Malcolm Garland [Ref. 4]. The model written for that thesis uses some of the maneuver

methodologies developed here to investigate the utility of using artillery fires to reduce mine density prior to the conduct of a minefield breach.

II. PROBLEM DISCUSSION

A. MINES

The military forces have been dealing with effective land mines since World War I. Since that time, human ingenuity and the motivation provided by numerous wars have resulted in continuously improved effectiveness for the military land mine. With the advances in the state of the art brought on by modern technology, land mines have become a very real threat to the success and even the survival of an attacking force.

Mines are unique among obstacles. Not only do mines fill the traditional role of an obstacle by delaying an attacking force, but land mines also have the capability to inflict other costs upon that attacker. Among them,

- lost time
- damage to equipment
- casualties
- lost ability to maneuver
- lost momentum
- logistical burden
- loss of morale.

Most of these costs are measurable to some extent, so a quantitative evaluation of the costs for a given minefield scenario can be made without the requirement to link these costs to a tactical result. Alternative ideas may be compared in terms of cost without consideration of what criteria are to be used for defining acceptable or

unacceptable. A computer model can be used to develop an estimate of the costs involved.

Mines pose a psychological as well as physical obstacle. A soldier that is worried about setting off a land mine will not be as aggressive as the soldier who is confident that a reliable path exists through the minefield. Morale is likely to suffer if no reliable means of dealing with the threat produced by land mines exists.

Avoiding a minefield may be worse than going through it. One use of mines is to deny a force the ability to maneuver in the mined area. In this way, minefields are used to channelize the attackers into prepared killing zones, allowing a small force, protected by minefields, to deflect an attacker into an engagement area designed to destroy it. The best course of action may be to breach minefields rather than to predictably attempt to go around them.

There are several broad categories of military mines used on the modern battlefield. These categories include, but are not limited to, chemical, antipersonnel, and antivehicular mines. Antitank mines are a subset of the antivehicular mine family. Such mines are designed to immobilize or destroy armored vehicles and their crews.

1. Mine Designs.

Antitank mines perform their function by either attacking the vehicles ability to move, resulting in what is called a mobility kill, or by the complete destruction of the vehicle, which is known as a catastrophic kill. A mobility kill is achieved by destroying one or more of the vehicles' vital drive components, usually breaking the

track, thereby causing the vehicle to be immobilized. With a mobility kill, the weapon system may still continue to function. A catastrophic kill will result in the entire vehicle and/or crew being destroyed.

Current production antitank mines may weigh up to 25 pounds, be contained in either plastic or metallic cases, and come in a variety of colors. Some mines are designed to be mechanically emplaced, while others are intended to be emplaced by hand. Antitank mines may be buried beneath the surface by either manual or mechanical means, or they may be surface scattered. Many countries have developed or are developing the ability to remotely deliver surface mines. Different types of mines are often mixed together.

Mines may be used in a variety of circumstances. Nuisance minefields are placed in order to hinder and disrupt an enemy. Hasty protective minefields are laid out to provide quick protection for a defending force. Deliberate minefields are obstacles integrated into the defensive plan and are usually of high density and emplaced in specific patterns.

2. Distinguishing Characteristics.

Conventional antitank mines can be distinguished by the type of engagement mechanism used and the method of fuzing that initiates the engagement. Engagement mechanisms include blast, self forging penetrators, and shaped charges. Methods of fuzing are numerous, but include single or multiple pressure pulses, command firing, magnetic influence, seismic vibrations, and motion detecting. Many mines also allow for the provision of antihandling devices to prevent removal.

Blast mines attack their target through blast caused by the detonation of a quantity of high explosive. These mines usually produce a mobility kill but a catastrophic kill is possible.

Self-forging mines engage their victims with an explosively shaped metal penetrator designed to defeat the thinner underside of a vehicles' armor and spray shrapnel and plasma throughout the inside of the vehicle, with the designed intent of igniting fuel, detonating ammunition, and killing the crew. The usual result of a successful engagement of this type is a catastrophic kill.

Horizontal-effect mines use a shaped charge to penetrate the thinner sides or top of armored vehicles, possibly resulting in a catastrophic kill. Depending on the type of sensors and fuzing used these mines may be emplaced a considerable distance from the expected engagement areas. These types of mines are frequently placed on the sides of roads and other vehicular avenues of approach, or they may be used to overwatch conventional minefields to prevent the use of breaching equipment.

Antitank mines are further distinguishable by their fuzing. There are three broad categories of antitank mine fuzes; track-width fuzes, full-width fuzes, and wide-area fuzes.

Track-width fuzes are usually pressure-actuated and require the wheels or tracks of the vehicle to pass directly over the mine. This type of fuze will normally produce a mobility kill since it will detonate directly under the tracks or wheels.

Full-width fuzes, such as tilt-rod, magnetic-influence or seismic fuzes, are designed to be effective across the entire width of the target. This type of fuze is usually employed in conjunction with a self forging or shaped charge warhead to produce a catastrophic kill when the mine is straddled by an approaching vehicle.

When a full-width fuze is activated under the wheels or tracks of a target vehicle, a mobility kill usually results because most of the energy is absorbed by the vehicle suspension.

Wide-area fuzes are used to enable mines to attack vehicles which do not pass directly over the mines location. The fuze may be as simple as a trip wire activating a horizontal-effect off road mine, or as complex as seismic activated, infrared searching, homing munitions. Such fuzes are normally placed on mines designed to produce a catastrophic kill.

B. BREACHERS

The only currently available solution to the problem presented by any buried or surface laid mine, regardless of its fuzing, is to detect and identify it as a mine and then:

- destroy it in place with explosives
- mechanically extract it or push it aside
- pull it out with a long rope or wire from a protected position.

For armor operations, the fastest and safest mine removal technique is to either detonate it from a distance, or mechanically extract or push aside any mine which happens to be in the path of the vehicle so that if the mine were to detonate it would do so with minimum danger to personnel and little or no damage to the vehicle. A

number of techniques and devices for accomplishing this action have been devised over the years. The most common and proven of these follow.

Full Width Plow. Full width plows are blade and tine assemblies designed to remove mines for the full width of the blade and spill the mines, as well as a significant amount of soil to either side of the breach lane. They are usually a permanent part of a specially designed engineering vehicle. Such vehicles are specifically designed with the horsepower required to push the blade through a variety of soil types.

Current plows may make single pass lanes up to five meters wide, and depending on soil type up to 40 centimeters deep. The tines are typically designed to remove mines 20 centimeters in size or larger. Plows may be particularly effective at removing surface laid mines, since the depth of cut need be only a few centimeters.

Full width plow vehicles tend to be slow. Their designs are optimized to provide the tractive forces necessary to move large amounts of soil. Being slow, they are particularly easy targets for the enemy forces typically overwatching minefields. Being breachers, they are given a high engagement priority by the defenders.

The designers of mines are also not without recourse when confronted with plows. Although the plow itself is usually quite sturdy, some of the mechanisms used to control the depth of cut are more vulnerable. Mines equipped with an antihandling device will go off when disturbed by the blade. The resulting blast, depending on the size and type of mine, may be sufficient to damage the blade or the more delicate control mechanisms, thereby rendering the blade less effective and more vulnerable to the remaining mines in its path. Area effect mines are not dependent on the close

proximity of the target for activation, and these types of mines, when added to the minefield, would be particularly useful in destroying the breachers.

Tank Mounted, Track-Width Plow. Track width plows consist of plow assemblies which mount directly onto tanks. A major advantage of using track width plows is that they may be made organic to the unit, and therefore be constantly available for use. These devices plow the area immediately in front of the vehicle tracks, uprooting and/or displacing mines to the outside of the vehicle's path. The plows are capable of removing land mines or booby traps which are surface laid or buried up to 15 centimeters below the surface.

A weighted chain assembly, sometimes called a "dog bone" is strung between the two plows, in order to activate (harmlessly in front of the tank) any tilt rod type mines encountered.

Track width plows have several limitations. The "dog bone" only clears tilt rod type mines from beneath the vehicles. Influence type mines which happen to lay between the track plows would not be removed by either the plows or the "dog bone". Tanks, while capable of using the plows, are not designed for this type of activity, so there are consequences with regard to the depth of plowing possible and the maximum speed at which the vehicle can plow. Damaged plows may cause the vehicle to be at least temporarily disabled until the damaged equipment can be jettisoned.

As mentioned before, a counter to the use of plows is the attachment of anti-handling devices onto the mines. Plows are vulnerable to mine explosions, and one or two may destroy the plow's effectiveness.

Flail. A flail is a rotating cylinder, mounted in front of the vehicle, with chains, sometimes tipped with weights, attached to the cylinder. As the cylinder rotates, the

chains or weights pound the ground in front of the vehicle. This action is intended to detonate or physically destroy any mines in the path of the flail. Exploding mines, unless very powerful, will at most destroy a single chain and weight. This type of equipment tends to be quite complicated mechanically and is generally not organic to maneuver elements.

Tank Mounted Roller. Rollers are heavy cylinders which are either rolled in front of the breaching vehicle, or towed behind it. A roller clears mines by detonating them with direct pressure. The roller is made of a material tough enough to absorb several detonations.

One counter to this breaching technique is a delay fuze timed to detonate a short period of time after activation, hopefully (from the mines owner's viewpoint) under the breaching vehicle. Another counter-counter measure would be a multiple pressure pulse fuze, which must be activated several times before it detonates its explosive charge, thereby allowing several vehicles to pass before detonating and blocking the lane. Rollers are usually used to 'proof' minefield lanes.

Line Charge. Line charges are used to detonate mines in the projected path of the breaching force. A rocket is used to tow a explosive filled hose across the minefield. The explosive in the hose is then detonated. The overpressure created by the explosion detonates pressure fuzed mines that are in the vicinity. Pressure mines equipped with a fuze which requires a long pressure pulse will not be affected. Influence mines are also not likely to be affected unless they are so close to the charge as to be physically destroyed by the blast.

Fuel Air Explosive. Fuel Air Explosives (FAE) use a similar technique as the line charge. An explosive vapor is created and then ignited. The resulting explosion is

very intense and intended to clear pressure fuzed mines in the area of the blast by creating a pressure pulse.

C. DOCTRINE

1. Countermine Operations

The taskforce commander, when in combat, can expect to be confronted by a variety of obstacles. These obstacles must be overcome to maintain the initiative and the momentum of the attack. The taskforce commander must quickly decide whether to bypass, breach, or force through the obstacle.

The obstacle should be scouted to determine if it is part of an occupied defensive position, and if so, the enemy strength and locations.

"Forcing through" is a term used to describe the tactic of ignoring a minefield and attempting to simply drive across it, using only the equipment on hand and with a minimum of preparation. Vehicles may travel in single file in an effort to reduce risk, however, casualties are an expected result of this technique. Forcing through a minefield obstacle is used only as a last resort. Forcing minefields can cause substantial losses of personnel and equipment. The urgency of the taskforce mission will be the deciding factor.

When an obstacle is encountered, it must be rapidly reconnoitered to determine if bypass routes exist and if such routes are covered by enemy fires. When possible, current doctrine calls for enemy minefields to be bypassed rather than breached since bypassing maintains the momentum and conserves critical countermine assets.

However, any decision to bypass must consider the possibility of the friendly units being channelized into kill zones.

If the decision to breach is made, then the type of breaching operation to be conducted must be decided. If time is important, then the unit may attempt a hasty breach. If time is less critical, or if the likelihood of a successful hasty breach is very poor, then the taskforce may conduct a deliberate breach of the obstacle.

The hasty breach is a tactical assault breach used when the momentum of the attack must be kept up. If engineer assets are currently attached or reasonably available, they may be used. Any organic breaching equipment will be prepared and utilized. The breach will usually be conducted while under enemy fire and because of mission critical time constraints, speed of execution is important. When the time available to begin the conduct of the breach of a minefield is not critical, a deliberate breach may be conducted using engineers and specialized breaching equipment.

Maneuver forces assigned the mission of breaching the obstacle will normally be organized into specific elements as part of the assault breaching plan. These elements are known as the breaching force, the assault force and the support force.

The breaching force has the mission of actually creating the lanes through an enemy obstacle system for the assault force to pass through. These forces are normally composed of engineers, scouts, and armor. If available and time permits, specialized breaching equipment is obtained and utilized. After the breach is completed, the breaching force is normally reorganized to assist the assault force.

The assault force has the mission to attack through the breach, penetrate the defense, and destroy the enemy. An assault force is normally built around a combined

arms tank unit. The assault force will attempt to stay within the boundaries of the lane created by the breaching force.

The support force includes all units providing close, continuous, overwatching fires to support first the breaching force and then the assault force. The support force normally consists of tanks; wire-guided missile systems; organic indirect fire elements; field artillery in close support; and chemical company (smoke) elements if available. The support force may be required to widen initial lanes created by the breaching force, or support another unit in that task with suppressive fires.

The assault force and then the support force provide suppressive fires as the obstacle is reconnoitered and the breaching force prepares to breach. If the decision is made to breach the obstacle, flank security must be provided.

The actions taken by the various elements in the conduct of a breaching operation follow:

- Support Force. The support force occupies overwatch positions in order to protect those elements moving through the obstacle. The support force will provide direct and indirect suppressive fires on the enemy. The support force will use smoke (pots, mortars, artillery, grenades) as appropriate to degrade enemy observation of the obstacle.

- Breaching Force. As an obstacle is encountered, the breaching force immediately occupies covered and concealed fighting positions; hastily coordinates specific tasks; and prepares equipment, demolitions, and routes to the obstacle. Once enemy fires are suppressed, the breaching force rapidly breaches the obstacle. The force then secures fighting positions near the far side of the obstacle as quickly as possible.

- Assault Force. As the breaching force is breaching the obstacle, the assault force prepares to attack through the obstacle. Once the breaching force secures initial positions on the far side, the assault force attacks through the obstacle and destroys enemy elements that may be able to place direct fires on the obstacle. Then it either continues the advance as the lead element for the attacking unit, or occupies hasty defensive positions as the support force passes through and takes the lead in the attack.

When a unit is attacking across open terrain, the minimum distance between lanes should be 250 to 300 meters. This keeps the enemy from blocking more than one lane with a single artillery concentration. The distance between lanes may be greater than 300 meters, depending on the commander's ability to maintain control of his force.

2. Mine Operations

Planning for the effective use of minefields as obstacles requires the consideration of numerous factors. Foremost must be the commander's plan of operations. For most mines to be effective as an obstacle considerable planning must go into the logistics of transporting the mines to the desired location and then installing them. Some mine types are quite bulky and require extensive logistical support as well as individual installation. Others are small and can be rapidly emplaced with mechanical devices, or be delivered from missiles, aircraft or artillery.

For the purposes of this paper, the only minefield parameter used will be the density of the minefield. Minefield density is used as a means of expressing the relationship between the number of mines emplaced and the size of the minefield. It provides an indication of the "effectiveness" or "difficulty" of the minefield as an obstacle. There are two ways of expressing minefield density: linear density and area density.

The linear density of a minefield is the average number of mines by type per meter of minefield front, regardless of the depth of the minefield.

Area density is a measure usually associated with scatterable mines. The measurement specifies the number of mines per square meter of minefield area.

D. DISCUSSION

This model currently only represents the use of plows to displace mines from the vehicle path. The only mines currently modeled are pressure, influence, and an antihandling fuzed mine which will be called a "contact mine". The model is designed for the easy addition of other mine types and breaching techniques.

One of the problems relating to the breaching of minefields involves tradeoff analysis between the survival of the breacher and the effectiveness of the lane the breaching vehicle creates.

The survivability of the breacher is a function of the time required to complete the breach lane and the effectiveness of the breaching technique. The time aspect of this function is dependent upon the type and amount of enemy overwatching fire assigned to the minefield. Assuming at least a token amount of overwatching fires, then the more time it takes for the breacher to make the lane, the longer it is exposed to the overwatching fires and the greater the likelihood of the breacher being disabled by those fires. Another consideration is that the longer it takes to complete the breach, the longer the defender has to react by moving in reinforcements and calling for supporting fires.

If the breacher is not effective at removing or disabling the mines in its path, the breacher may be destroyed by the minefield itself.

The usability of the resulting lane by trailing vehicles is also a function of the effectiveness of the breacher in eliminating the threat posed by the mines as well as the navigation accuracy of the vehicles. Trailing vehicle speed is also a factor, again depending upon the type and amount of overwatching fires present.

The issue comes down to the time required by the breacher to accomplish the breach versus the width of the breach lane. The time to accomplish the breach is an important factor because it directly impacts on the survivability of the breaching vehicle. Of course other factors also impact on the breaching vehicle's life expectancy. These include the amount and type of overwatching fires, vehicle armor placement and thickness, minefield characteristics, mine clearing effectiveness, etc. Intuitively, since the more dirt we are pushing, the slower we push it, we would want to plow a lane as narrow and as shallow as feasible.

The flip side of the problem is that the breach lane must be wide enough for trailing vehicles to "safely" negotiate the path. Realize that a single mine encounter may result in the lane being blocked. We cannot simply choose a width to accommodate the widest vehicle (probably a tank) but we must also allow for the fact that combat vehicles are not known for their precision handling, just as most military drivers are not known for their precision driving. A certain amount of "navigation error" is to be expected, and the lane must be wide enough to allow for this error.

Speed through a minefield, as it impacts on survivability, is important only when the minefield is covered by direct or indirect fire. When a minefield is covered by fire, the longer a vehicle is in the field, the greater the chance of immobilization. The movement speed through the minefield is initially a problem relating to how long it takes the breacher to make a lane. Non-breacher vehicles would stay out of the

minefield until a lane has been made, or all breachers have been disabled. Upon completion of the lane, other vehicles would attempt to pass through. If they are immobilized, then the lane may be blocked.

III. METHODOLOGY

This thesis presents several methodologies, along with their corresponding SIMSCRIPT coded algorithms, which are designed to provide a basic framework for future minefield modeling efforts. The algorithms are intended to serve as a foundation for more ambitious efforts directed toward high resolution modeling of mobility and countermobility scenarios. The model currently has only one category (plows) of breaching equipment represented. The methodologies presented are designed to model basic minefield related activities to include:

- navigation by an assaulting force
- discrete mine-vehicle encounters
- mine displacement by plows
- overwatching fires by the defenders.

The methodologies and algorithms presented are flexible enough to apply to many different scenarios with only minor modifications. The model itself is designed for easy expansion and modification to include the addition of other types of breaching equipment, vehicles, mines, and obstacles.

The approach taken in presenting these methodologies is to first discuss the real life mechanics of the situation and the significance of the attributes which are to be modeled, and then to give a description of how the attribute is to be captured in the program code. Chapter IV will go into more detail as to how the program actual

implements the algorithms and in Chapter V an example of how they can be used will be presented.

A. VEHICLE NAVIGATION

As a vehicle moves over an area, it casts a shadow or a "footprint" on the ground. In this paper, the actual area which passes under the tracks or wheels of a vehicle will be called the "pressure footprint" and that area which passes under any portion of the vehicle will be labeled the "influence footprint". Note that the pressure footprint is a subset of the influence footprint.

The individual footprints for multiple vehicles moving over a piece of terrain will vary as a result of several factors. If the taskforce consists of a variety of vehicle types, the physical dimensions of the different type vehicles will result in different footprints. The size of these footprints can be determined by examination of the vehicle specifications. Another less predictable cause of variation is what this paper will call "navigation error".

It is essentially impossible for any two vehicles to have precisely the same footprints, even if they have the same physical dimensions. This is because of inaccuracies inherent in the steering mechanisms of the vehicles and because of the human element introduced by the drivers of each vehicle. The human element may be particularly apparent at night. The resulting differences in footprints due to the mechanical and human inaccuracies is the "navigation error".

The situation may arise where a vehicle is unsuccessful in passing through a minefield even though it is attempting to follow the same path as a successful vehicle. Although there are multiple reasons why this might happen, one possible explanation

is the difference in the respective vehicle footprints. Specifically, the trailing vehicle did not have exactly the same footprint in its traversal of the lane. Even small deviations from the lane may be hazardous. Figure 3-1 shows the resulting mine densities after a lane has been made by a full width plow and by a track width plow. Note that the only area that might be considered safe is that portion of the lane that makes up the pressure footprint of the breaching vehicle. The areas to either side of the lane are now more dangerous than they were originally, since they hold not only their original complement of mines, but also those mines displaced by the plows. It

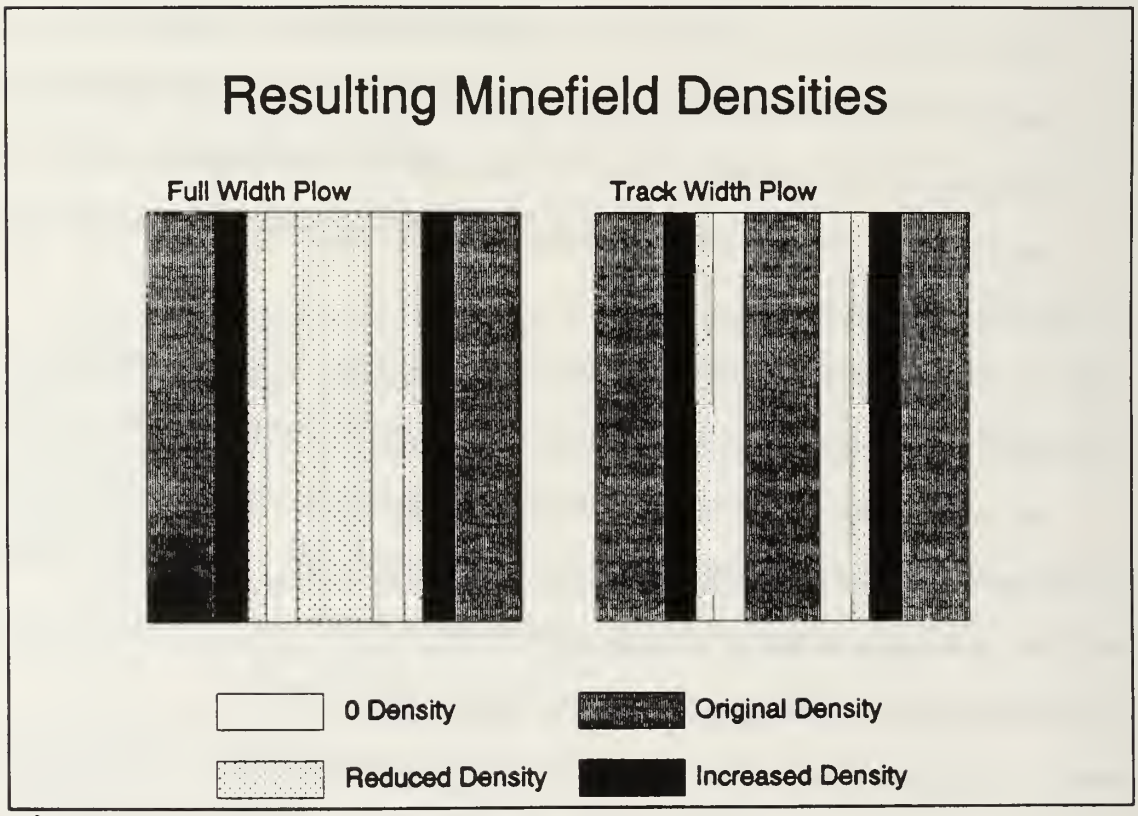

Figure 3-1

is possible that the displaced mines will have a decreased lethality, however the worst case assumption is that any displaced mine is still capable of functioning as designed. At any rate, at least some of the displaced mines would still be viable, so the actual mine density of the lane shoulders has increased. Note also that the lane itself may still contain mines which were either buried deeper than the plow was set to dig, or slipped through the tines on the plow, or rolled back into the lane from the side spill.

This model makes provision for "navigation error" in the traversal of the minefield by multiple vehicles. The amount of navigation error used is a function of certain values input by the user. The methodology used is as follows.

The intent is for each taskforce vehicle to have a unique movement plan consisting of a user specified number of navigation checkpoints. Each vehicle will move from checkpoint to checkpoint, in a straight line between the adjacent checkpoints, following it's own unique (in the **x** dimension) movement plan. Each movement plan will have the same number of navigation checkpoints. Vehicles are assumed to have the desire to follow in the footprint of the lead vehicle, therefore, all deviation is measured from the actual centerline of the path made by the lead vehicle.

The **y** axis is arbitrarily designated as the center of the breach lane through the minefield. Since the first vehicle is not attempting to follow a lane, but is in fact making the lane, it is assumed to have no navigation error. It's movement plan consists of a series of navigation checkpoints located on the **y** axis.

The total distance, from start point to finish point, to be traveled by the taskforce is divided into a user specified number of equal intervals. The **y**-coordinates of the endpoints of these intervals become the **y**-coordinates of the movement plan

checkpoints for all taskforce vehicles. That is, the y-coordinates of respective checkpoints will be the same for all members of the taskforce.

For each checkpoint an x-coordinate is determined through use of a normally distributed random draw. Figure 3-2 depicts this procedure.

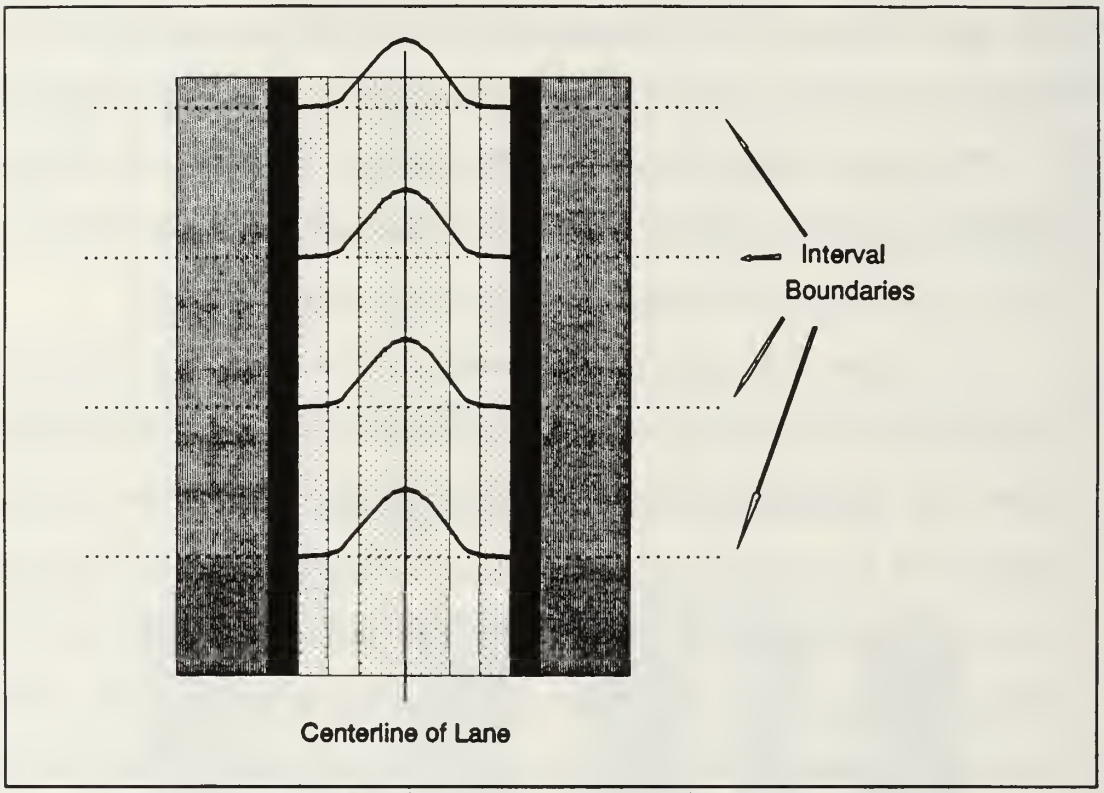

Figure 3-2 Determining the X-coordinate

The mean of the distribution is assumed to be zero, meaning that a vehicle is assumed to be as likely to deviate to the left as to the right. Should the user wish to modify this assumption, it is an easy matter to change the distribution parameters or even the distribution itself. The standard deviation used is a user input. The default value used for standard deviation is one meter.

A normal distribution is used, because it is assumed (and would certainly be the desire of the vehicle crews) that the vehicles will tend to follow the center of the lane. Small deviations are not only likely, they are unavoidable. Large unintentional deviations are unlikely. Deviations are equally likely to occur on either side of the breach lane and there is no reason to believe that the magnitude of the deviations will differ as a result of on which side they occur, so the resulting distribution should be symmetric.

The process of generating **x**-coordinates is continued until each vehicle in the taskforce has a series of checkpoints, unique in the **x**-dimension. An example of a resulting movement plan for one vehicle can be seen in Figure 3-3.

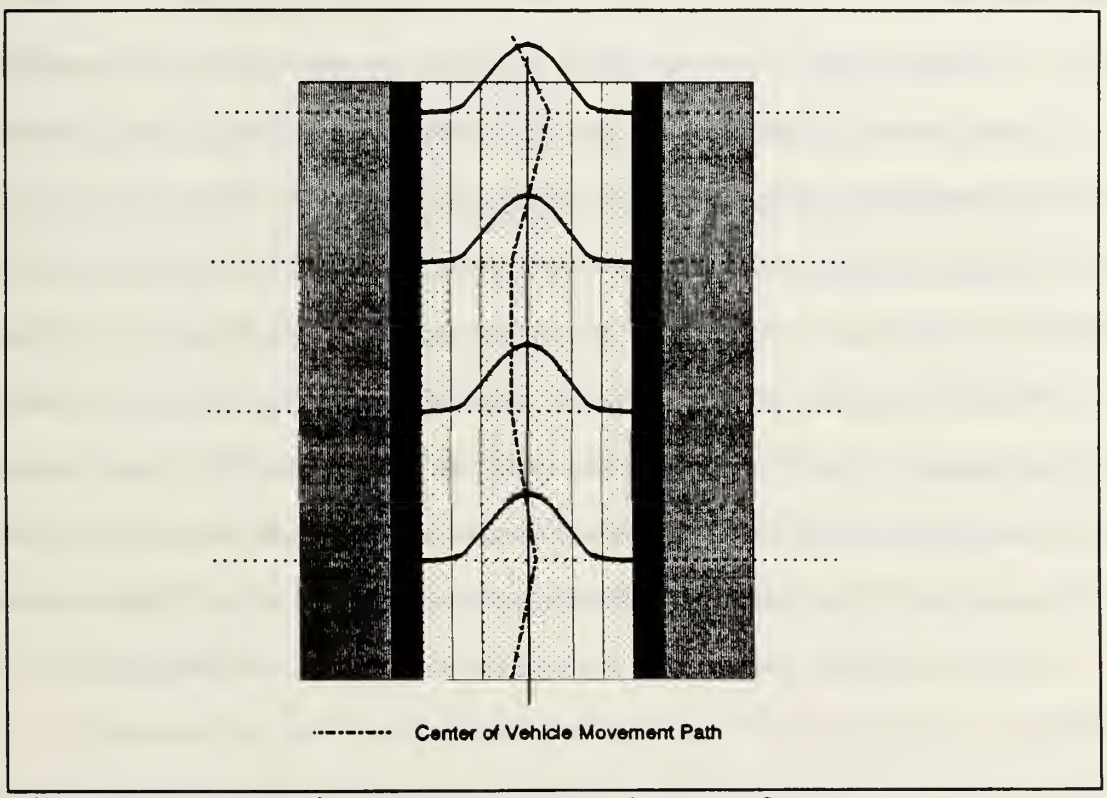

Figure 3-3 Vehicle Movement Path Example

Once the **x** and **y** coordinates of all of the navigation checkpoints for a vehicle have been determined, the equations for the lines which connect adjacent checkpoints are determined. This is done by first determining the slope, m, using the equation;

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad , \quad x_2 \neq x_1 .$$

The **y**-intercept, b, is then determined by using one of the checkpoints at either end of the line, the previously determined slope, and the equation,

$$b = y - mx .$$

Each vehicle now has a unique movement plan governing its movement which also allows for navigation error. The vehicle will be moved along the line that connects its checkpoints at the rate indicated by the vehicle's speed. A pointer to the currently used line equation indicates to the program routines the appropriate slope and intercept values to be used for calculating the vehicle's location at any given time. When a checkpoint is reached, the pointer is moved to indicate the next line equation to be used. This process continues until the vehicle completes its movement or is immobilized. Note also that the actual use of straight line segments to describe the movement of tracked vehicles closely approximates the reality. Of course, in reality, tracked vehicle do not move in straight line segments of uniform length.

B. MINE ENCOUNTERS

As described above, each vehicle has a unique movement plan for traversing the minefield, described by checkpoints and the equations for the lines which connect those checkpoints.

Each mine in the minefield has a type, a radius and an **x** and a **y** coordinate, which describes the unique location of that mine in the minefield. Type is used to identify the fuzing method used. Currently the model plays either pressure, influence, or contact (antihandling) mines.

Pressure fuzes require the tracks of the vehicle to actually pass directly over a portion of the mine before the mine will detonate. Influence fuzes only require that a portion of the vehicle pass over the mine, making them effective for the entire area of the vehicle as it moves over the field. Contact fuzes are similar to pressure fuzes in that they require some actual form of contact, either directly with a portion of the vehicle or by earth compressed by the weight of the vehicle, in order to function. They are distinguished however, by the amount of pressure required. Contact mines require only to be disturbed in order to activate. They are intended to destroy breaching equipment, detonating when a plow (or anything or anybody) attempts to move them. While this characteristic makes them a threat to most breaching techniques, such mines are generally quite vulnerable to breaching techniques that rely on explosively generating shock waves and overpressure to clear pressure mines from the minefield.

Encounters are calculated by first screening the entire list of mines to determine which ones are within a reasonable distance from the vehicle. By doing this we eliminate unnecessary calculations and speed up the simulation.

Then, for each candidate mine, the equation of the line that passes through the center of the mine and is perpendicular to the vehicle path equation is calculated. First, the slope of the perpendicular line is determined through use of the theorem that two lines are perpendicular if and only if,

$$m_1 * m_2 = -1$$

where m_1 is the slope of the vehicle path and m_2 is the slope required for a line perpendicular to the vehicle path.

The equation for the perpendicular line is found using the point slope formula,

$$y_1 - y_2 = m_2 (x_1 - x_2) .$$

The intersection of the perpendicular line and the vehicle path is then determined by solving the two line equations simultaneously. This point is where the vehicle has a potential contact with the mine. Figure 3-4 illustrates the geometry of this procedure.

To determine if contact actually takes place, the distance between the center of the mine and the intersection point is calculated. Adjustments are made to account for mine radius and the fuzing mechanism used by the mine in question to determine if a contact will actually take place. If a contact will take place then the distance between the current location of the vehicle and the intersection point is calculated.

If an actual contact is possible, the mine identification and distance from the vehicle are recorded and compared with the current closest mine. If this mine is closer, its identification and distance become the current closest. The next candidate

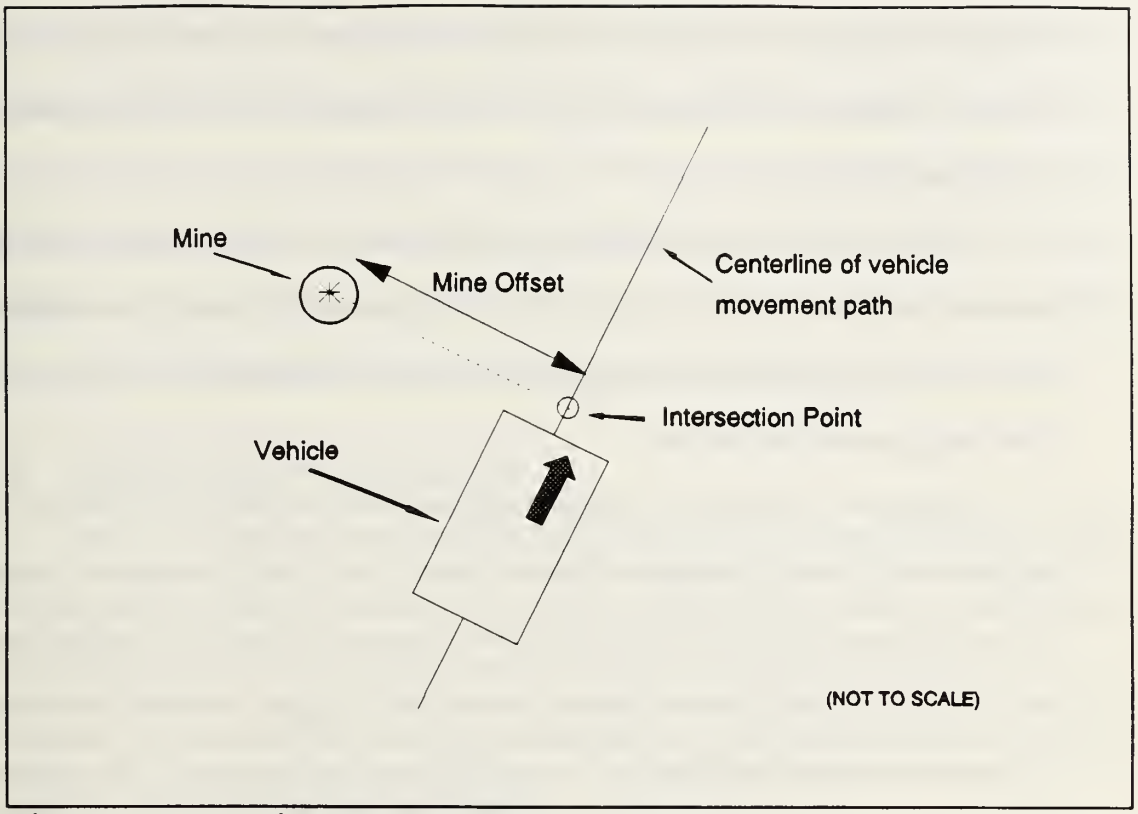

Figure 3-4 Mine Encounters

mine is checked. This continues until all candidate mines are examined. The mine identification and the distance the vehicle must travel to the closest mine with which contact will occur are then returned to the calling routine.

C. MINE DISPLACEMENT

Mine plows are not designed to actually destroy the mines that they encounter. The mine plow performs its function by displacing any mines encountered, as well as a substantial amount of dirt, to either side of the breach lane. To perform adequately, the plow must actually remove the mines from the lane and place them far enough to either side that they are no longer a threat to vehicles navigating within the lane.

Plows should be particularly effective against surface laid mines since they do not have to move much soil, thereby allowing the plow to maintain a reasonable speed while working on the lane.

As Figure 3-1 previously illustrated, while the area of the actual breaching lane becomes safer, the areas to either side of the lane actually become more dangerous. There is also a small but finite possibility of displaced mines rolling back into the lane after being moved by the plow.

Each plow type modeled has several parameters attached to it. Of particular importance is the effectiveness of the plow which is an attempt to capture the degree of thoroughness with which the plow removes mines in its path. An effectiveness of one indicates that the plow will remove every mine in its path, assuming that the mine is buried no deeper than the depth at which the plow is operating. An effectiveness of less than one indicates that a percentage of mines encountered will be missed by the plow, possibly to be encountered by the pressure or influence footprints of the plow itself or the trailing vehicles.

The methodology used to capture the consequences of plowing a lane requires that non-contact mines encountered by a plow be moved to the side of the lane. A new **x** and **y** coordinate is calculated for the mine based upon the location of the edge of the plow and a probability distribution. Should the mine be of the contact type, then further calculations must be made to determine the result of the encounter. That is, what degradation of effectiveness was inflicted upon the plow.

The distribution currently used is normal with a mean of 1.5 meters to the left or right of the plow edge and a standard deviation of 1 meter. The determination of which side of the plow the mine is displaced to is simply a matter of on which side of

the centerline of the plow the mine falls. Since most plows are shaped somewhat like an inverted "V", mines located to the left of the centerline naturally are displaced to the left and those on the right to the right. The choice of the normal distribution and the parameters chosen are purely arbitrary and may be easily changed should the user desire. It should be noted though, that the choice of mean and standard deviation allow for a small possibility of the mine coming to rest within the lane.

D. OBSTRUCTION AVOIDANCE

When a vehicle becomes disabled during the course of a minefield breach, either by overwatching fires or contact with a mine, it becomes part of the overall obstacle breaching problem. If the vehicle is negotiating the "cleared" lane when it becomes disabled, then the vehicles following it must take one of the following three actions in order to continue using the cleared lane.

First, the disabled vehicle may be extracted from the minefield by attaching cables or a tow bar to it and pulling it either the rest of the way through the minefield or back out over its entry path. This will result in considerable delay and increased exposure by all concerned to overwatching fires.

The trailing vehicles may attempt to push the disabled vehicle, either through the remainder of the minefield, or to one side of the cleared lane. Pushing a disabled tank through a minefield has sometimes been considered as a field expedient method of breaching minefields. There are numerous potential problems with this technique. If a track has been lost, the disabled vehicle will not move in a straight line. If the damage is extreme, then the disabled vehicle may not be moveable at all without special equipment. If the disabled vehicle is on fire, there is a danger of explosion and

damage to vehicles in the proximity. A burning vehicle may also serve as a beacon and aiming point to enemy elements overwatching the minefield. The "pushing" vehicle may damage its own track and suspension components in the effort. The pushing vehicle may stray from the path in the attempt and also encounter a mine. If there are still crew aboard the disabled vehicle, either unable or unwilling to exit, pushing the vehicle into uncleared areas subjects them to further risk.

A final option is to go around the obstruction formed by the disabled vehicle, leaving the cleared lane and entering uncleared portions of the minefield for the

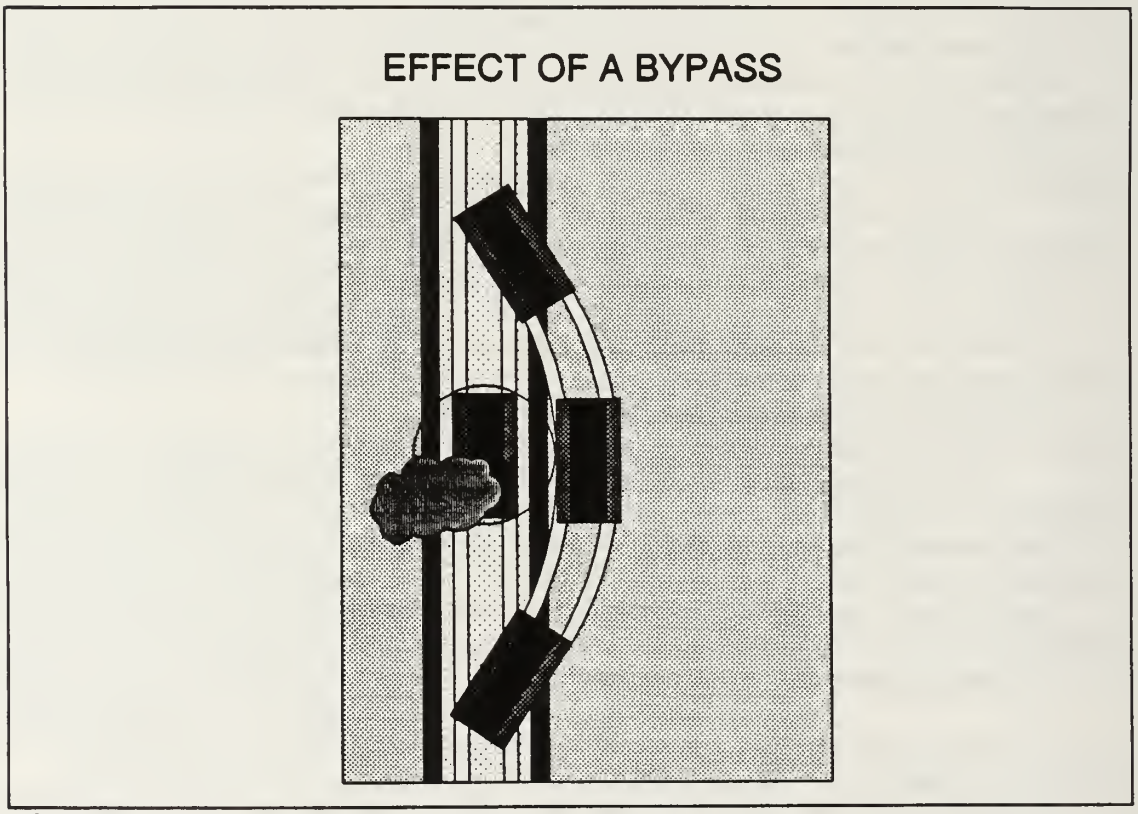

Figure 3-5

distance necessary to bypass the obstruction. The bypassing element would then reenter the cleared lane on the far side of the obstacle. Figure 3-5 depicts this action. Note the greater risk of mine encounter resulting from passing through the edges of the cleared lane and the uncleared portions of the minefield. If the bypass is successful, a new 'cleared', but unmarked, lane results. If unsuccessful, that is, if the bypassing vehicle is itself disabled, either by the minefield or by overwatching fires, then the lane obstruction becomes larger.

Bypassing vehicles will want to leave the cleared path for the shortest distance necessary in order to bypass the obstruction. This shortest path should include as little as possible of the higher mine density shoulders of the cleared lane.

Bypassing an obstruction in the lane may be a high risk technique, particularly if the mines have been laid out in belts and the obstructing vehicle has been disabled by a mine. In this case there would be a high probability of additional mines being to either side of the disabled vehicle. This situation may hold true for the entire width of the minefield.

The methodology presented in this model addresses only the tactic of bypassing a disabled vehicle. If the options of recovering or of pushing the disabled vehicle are desired those routines will have to be developed as an improvement or addition to the model.

Whenever a taskforce vehicle is disabled by either mines or overwatching fires, an obstacle is created. This obstacle entity will have the same physical dimensions and location as the disabled vehicle. The disabled vehicle will have its status changed to "inactive" and will be ignored for the remainder of that particular model run. The location and size of the newly created obstacle will be added to the obstacle list.

Every time an active vehicle is changed to an inactive status due to a mine encounter or overwatching fires, the event calendar for the simulation must be purged of all pending encounter events. When this step is complete a new next encounter event for each active element is redetermined to prevent the occurrence of "invisible obstacles". An invisible obstacle is one which is ignored by the active vehicles in the simulation. They are ignored because they were created after the trailing vehicles had already determined their next encounter, which should now be an obstacle encounter, but since at the time of determination the obstacle did not exist, it was not considered.

All obstructions, and the vehicles that are attempting to bypass them, are approximated as circles by computing the two dimensional surface area of the obstacle or vehicle and then determining what radius a circle would have to have to encompass that area. This is done to allow an efficient algorithm which uses simple geometric constructs. The costs of using such an approximation are a relatively small loss of accuracy in the **x** and **y** coordinates of the vehicle's sides and rear. The geometry used for determining mine encounters does not use the circle approximation, but rather uses the appropriate vehicle dimensions.

When an active vehicle is determining what its next encounter is to be, the first step is to determine its current checkpoint interval and corresponding movement equation. The obstacle map is then referenced and obstacles which lie at least in part within that interval are examined as potential obstacle encounters. The next step taken is to determine the path a vehicle would have to take in order to bypass the obstruction. Logically the path should be as short as possible in order to minimize the vehicles exposure to the uncleared portions of the minefield and return to the cleared path as soon as possible. Given the circle approximation, the shortest possible distance

would be a semi-circle in the direction of the shortest path around the obstacle. Since tracked vehicles typically move in straight line segments rather than semi-circles and since the calculations required to interface the quadratic circle equation with the linear equations used by the rest of the program would be extensive and run time expensive, a straight line approximation is used. The resulting path around the obstacle is played as an equilateral hexagon.

Figure 3-6 illustrates the hexagon bypass route as overlaid on an circular obstruction. The numbers **1** through **6** denote the corners of the hexagon and the letters **A** through **F** identify the six sides. Note that all six sides of the hexagon are tangent to the circle at their midpoints. The center of the circle has the same location as that of the obstruction and the radius of the circle is equal to the radius of the obstruction plus the radius of the bypassing vehicle.

The coordinates for each of the six corners of the hexagon are calculated as follows. Following along with Figure 3-6, the symbol **Ω** is used to denote the **x** coordinate of the center of the obstacle. The symbol **δ** is used to denote the **y** coordinate. The character **r** denotes the radius of the bypass circle. The value of **r** is determined by adding the radius of the obstacle to the radius of the bypassing vehicle. **O** denotes the center of the circle.

Construct vertical lines at **+ r** and **- r** units from the center. Observe that side **B** of the hexagon is a segment of the vertical line located **r** units in the positive **x** direction from **Ω** and that side **E** of the hexagon is a segment of the vertical line located **r** units in the negative **x** direction. With the top of the diagram representing 0°, construct vectors from the center of the circle directed outward at 60°, 120°, 180°,

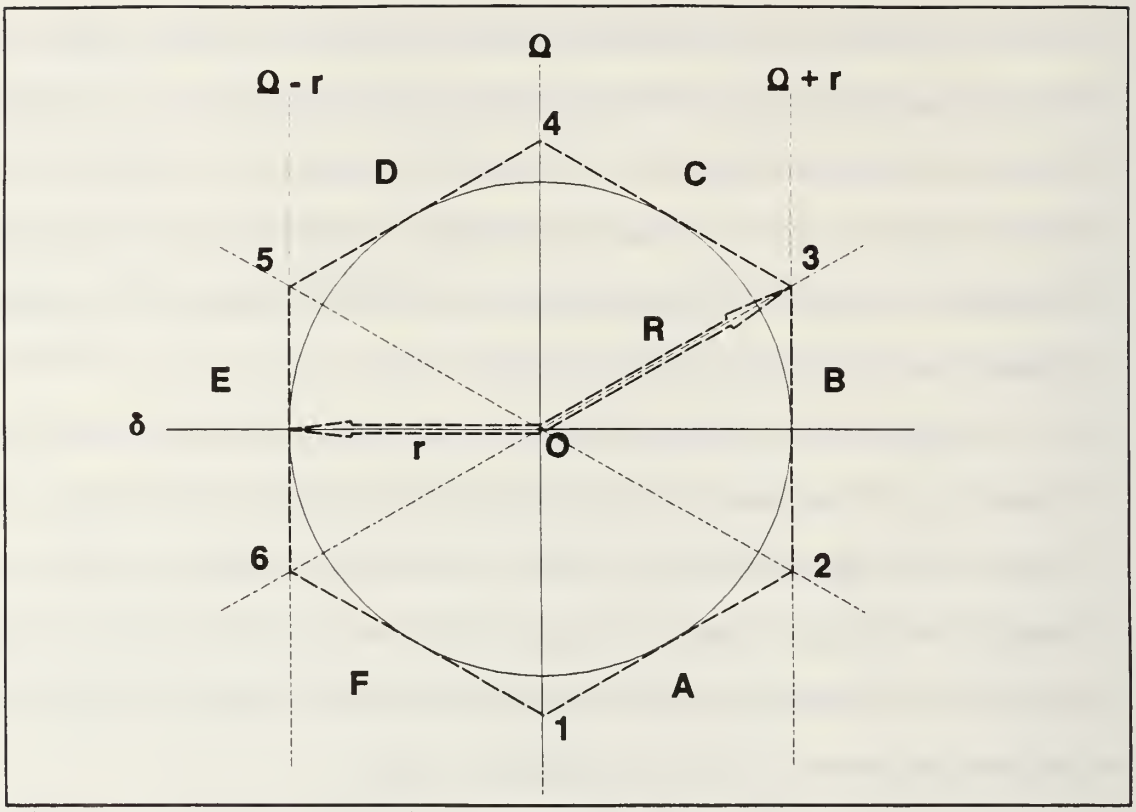

Figure 3-6 Obstruction Bypass Path

240° and 300°. Corner **2** is located at the intersection of the 60° vector (labeled **R** on Figure 3-6) and the vertical line at + **r**. If a horizontal line is constructed through the center of the obstacle (labeled **δ** on Figure 3-6) it can be seen that the triangle resulting from the intersections of the **δ** line, + **r** vertical line, and **R** is a 30-60-90 degree triangle. From this observation we now know that the length of side **B** is

$$B = 2 * \frac{1}{\sqrt{3}} * r$$

and that the length of the distance from the center to the intersection of the 60° vector and the **+r** horizontal line is

$$\frac{2}{\sqrt{3}} * r.$$

Symmetry and similar calculations are now used to determine the coordinates of all of the remaining corners. The edge of the resulting hexagon will be used as the centerline of the path a vehicle must take in order to bypass the obstruction in the lane. The equations for the lines connecting the corner points are generated in the same manner as the lines connecting the navigation checkpoints. A slight modification must be made for the two vertical sides, **B** and **E**, since the slope of a vertical line is undefined. An approximation is made by making the slope of any vertical line equal to an extremely large number.

Variation (navigation error) can be added to vehicle footprints during a bypass by adding a small distance, randomly generated, to the value calculated for the obstruction radius.

The point at which a vehicle will exit from its original movement plan path and enter the bypass path is the point where the current movement path line and the line describing either side **F** or side **A** of the hexagon intersect. Actually, the movement plan equation will intersect both lines since none of the three lines is parallel to either

of the others. The actual intersection of interest is the one which occurs on the hexagon itself.

The point where the vehicle regains the original movement path is called the "reentry point". The **y** location of corner **4** of the hexagon will be compared to the **y** locations of the navigation checkpoints to determine the boundary condition. That is, the interval where point **4** lies is the farthest interval the vehicle may reenter into from this bypass. The vehicle may possibly reenter in an earlier interval. All valid movement plan equations, the set of which consists of the line equations describing any movement plan segments between and including the movement plan equation used when the bypass path was initiated and the movement plan equation used within the boundary interval, will have their intersections with the lines describing sides **D** and **C** of the hexagon calculated.

The **y** coordinates of the intersections will be checked against the **y** coordinates of the checkpoints to which they correspond. If the **y** coordinate is valid, (i.e., if it falls within the appropriate interval) then the **x** coordinate will be checked against the **x** coordinates of the intersections. The first valid intersection point will be used by the vehicle as the reentry point.

As mentioned earlier, vehicles are exposed to greater risk of mine contact when bypassing, which would indicate a good possibility of disabled vehicles being in close proximity to one another. Additionally, if numerous vehicles are disabled the calculations for bypassing become very complex. To reduce the complexity, obstructions are combined when they occur within a user specified distance from one another. This simplification is justified by the observation that there is unlikely to be

sufficient distance between the disabled vehicles to allow a bypassing vehicle to pass between two vehicles, one of which was disabled as a result of an attempted bypass.

The combination of obstacles is accomplished by computing the centroid of the combined obstruction using the equations

$$x = \frac{(x_1 + x_2)}{2}$$

where x is the **x** coordinate of the combined obstruction and

$$y = \frac{(y_1 + y_2)}{2}$$

where y is the **y** coordinate of the combined obstruction. The radius of the combined obstruction, r_c, is

$$r_c = \max (D_{c1} + r_1 , D_{c2} + r_2)$$

where D_{c1} is the distance from the center of the combined obstacle to the center of obstacle 1, D_{c2} is the distance from the center of the combined obstacle to the center of obstacle 2, r_1 is the radius of obstacle 1, and r_2 is the radius of obstacle 2. Figure 3-7 illustrates the geometry of these calculations.

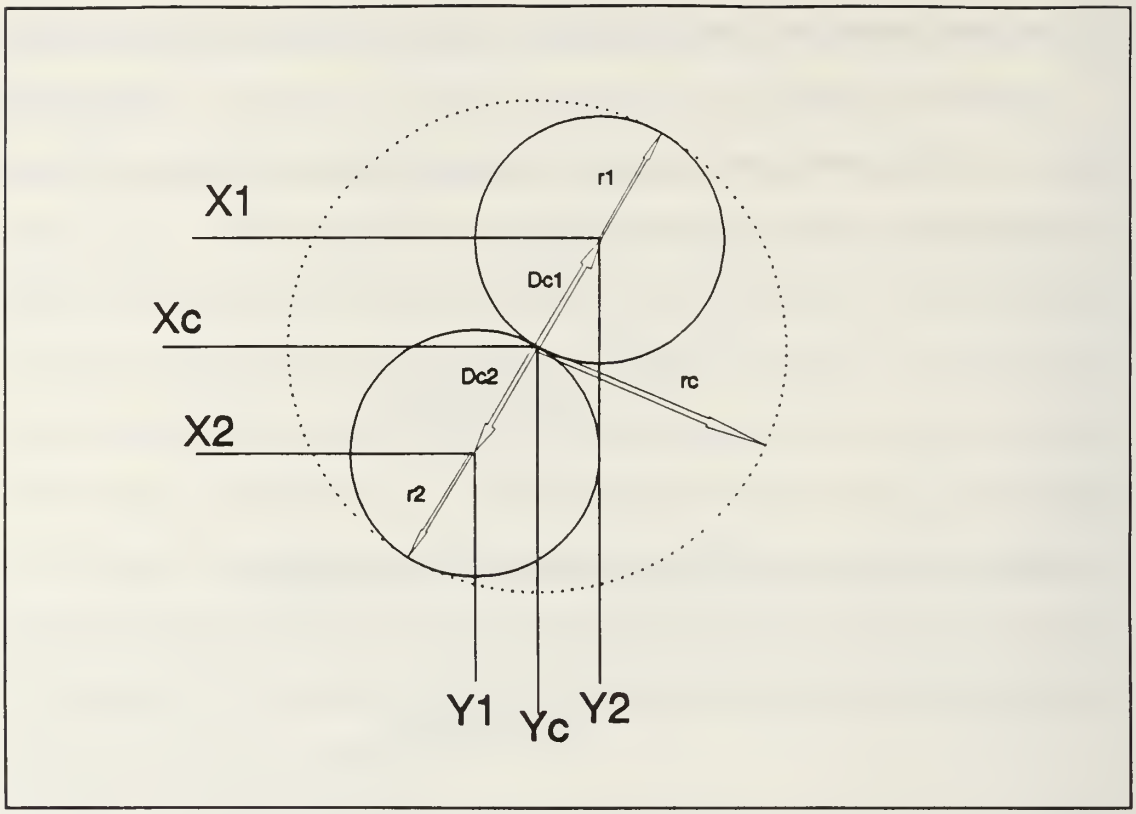

Figure 3-7 Obstruction Consolidation

As the simulation progresses, multiple obstruction combinations may take place. For this reason, it is necessary to keep track of how many obstructions have been combined into a particular combined obstruction. When the centroid of a new obstruction being added to a combined obstruction is being calculated the **x** and **y** coordinates of the combined obstruction must be weighted by the number of obstructions which make up the combined obstruction. The equation for the ***x*** component is

$$x = \frac{(N * x_c) + x_2}{N+1}$$

and the *y* component is

$$y = \frac{(N * y_c) + y_2}{N+1}$$

where *N* is the number of single obstructions making up the combined obstruction.

E. OVERWATCHING FIRES

Unless being used as a nuisance, or as harassment, minefields are doctrinally covered by fire. Actual military experience has indicated that a minefield covered by fire is more dangerous to the attacking force than the combined effect of the mines and fires, each inflicted in isolation of the other. In other words, there appears to be a synergism between the mines and the overwatching fires. The whole is greater than the sum of the parts.

As a possible explanation for this effect consider that if given unlimited time, mines can be dealt with in reasonable safety. The risks from the mines goes up with the decrease in the time available to deal with them. Overwatching fires serve to make the environment of the minefield increasingly lethal over time, regardless if the vehicles are moving or not. Therefore the combination of the increased lethality of the minefield as a function of reducing the time available to traverse it, and the increased lethality of overwatching fires as a function of increasing the time exposed as a target, results in greater casualties to the attacker.

The techniques for dealing with incoming fire generally involve movement. In the case of indirect fires, the goal of an attacking force is to move out of the area

where the fires are impacting. In the case of direct fires, an attacker will want to take cover and return fire, or use maneuver to close with and destroy the enemy. In either case, the attacking force is greatly hampered if it is attempting to deal with incoming fires while at the same time negotiating a minefield.

1. Direct Fires.

The model allows the user to enter the number of red overwatching direct fire elements as well their rate of fire, and a separate factor which combines detection, acquisition, and engagement times. This factor is actually the β (mean) parameter of an exponentially distributed random draw for representing a time period to be added to the minimum time required to prepare the system for firing (rate of fire). A direct fire event is then scheduled to occur after this duration.

Upon the actual occurrence of the direct fire event, the target of the shot is determined. An algorithm based upon the number of target systems actually in the minefield is used to determine the allocation of shots. The user selects the proportion of fires to be allocated to the first of **n** vehicles in the minefield. The remaining proportion is equally divided among all the other vehicles currently in the minefield using the relationship:

$$\begin{array}{c}\textit{Amount of Fire}\\ \textit{directed at each}\\ \textit{blue vehicle, 2}\\ \textit{through N}\end{array} = \frac{(100 - \%\ \textit{fire against Pos1})}{N-1}$$

Table 3-1 illustrates a possible series of proportions for a fire allocation scheme. The user can allow disabled vehicles to be included in the total count of vehicles in the minefield if so desired.

TABLE 3-1 FIRE ALLOCATION SCHEME

# of vehicles (n) in minefield	1	2	3	4	5	6	8	8	9
% of fires directed against Vehicle 1	100	80	70	60	50	50	50	50	50

After the target is determined, a table lookup for the probability of kill, p_k, given a shot for this particular weapon/target combination is performed, a random draw conducted, and the appropriate result carried out. If the vehicle is disabled it becomes an obstruction to trailing vehicles. If the vehicle survives the engagement it will continue to move. The firer identification, the intended victim, the time and location of the event, and the result can be recorded as output data.

When the direct fire event for the red overwatching vehicle has been completed, another is scheduled to occur at a time (time to prepare round to fire) + (exponential delay) in the future.

2. Indirect Fires.

This version of the model currently does not support indirect fires.

IV. MODEL DESCRIPTION

The methodologies described in Chapter III have been coded into an event driven minefield simulation model. This chapter contains a discussion of the assumptions under which the model was developed, a description of the model inputs and outputs, and a synopsis of how the model works. The descriptions of the model workings are of necessity fairly high level, as the model is quite large and the minute details are not the topic of this thesis. Documented source code listings can be found in Appendix C.

A. ASSUMPTIONS AND MODEL LIMITATIONS

This model is intended as an implementation of the minefield interaction methodologies previously discussed. The purpose of the model (and the methodologies) is to duplicate some of the outcomes of real world interaction between vehicles, breachers, overwatching fires, and mines. Several assumptions and simplifications have been made which have obvious implications with regard to transferring the model results to real world applications.

1. Uniform terrain

A major assumption is that the terrain to be traversed is uniform in elevation and in all applicable soil characteristics. A modeled plow, once its movement velocity and plowing depth have been determined, moves at a constant speed unless a status change is imposed as the result of an event. In the real world, terrain and soils are rarely this predictable, and in the event that they were, vehicle operators and plow control mechanisms are unlikely to maintain either constant speed or constant plow

depth. The impact of this assumption is that the variability in both breaching time and plow effectiveness may appear smaller than what would be expected.

2. Constant vehicle speed

Vehicles in the model move at constant speed throughout the simulation run. As a result there is none of the "accordion" effect between the vehicles that is nearly impossible to avoid in real world situations. Active model vehicles never get closer to each other than the interval input by the user during the model initialization. The one exception to this statement occurs when plowing vehicles are set to operate at a speed dictated by plow width and depth. In that case, the model user must either adjust the interval parameter which describes the initial interval between the plow(s) and the follow-on vehicles, or adjust the velocity of the trailing force to insure collisions do not occur as the result of the trailing force running over the plow(s). The model currently does not prevent nor react to active vehicles coming into physical contact.

The assumption of constant vehicle velocity greatly simplifies the movement calculations and is probably of little impact if the actual real world vehicle interval is anticipated as being relatively large and care is taken in the selection of the interval between plows and trailing vehicles.

3. Obscuration

Obscurants are not explicitly modeled. Since any real world opposed breaching operation is likely to be intensely supported with smoke operations, this is a large simplification. Some of the effects of obscuration, in particular, vehicle navigation and

target acquisition, can be captured by careful selection of the parameters used in the navigation and overwatching fire algorithms.

4. Breacher Types

The only breaching technique currently modeled is plowing. Since numerous different breaching systems exist, some of which were mentioned in Chapter II, this is a fairly large simplification of reality. Since the model is only intended to lay the groundwork for a high resolution minefield maneuver model, this shortcoming may be corrected at a latter date.

5. Mine Types

Only three types of mines are modeled, and all three types are of a fairly standard, unsophisticated nature. Many of the more interesting mines under development and in some cases currently in use, are very sophisticated and use state of the art technologies and exotic engagement and fuzing mechanisms. The maneuver methodologies in this program will still be applicable, but the code for capturing the fuzing and mine engagement mechanisms for these weapons will be left as a future enhancement.

6. Obstruction Types

Mobility obstructions, and the vehicles that must negotiate around them, are assumed to be of circular geometry. The reality is that obstructions come in a variety of shapes and sizes, and that vehicles are almost never circular. However, tracked vehicles do tend to move in straight line segments, and the assumption of a circular geometry allows the relatively simple calculation of a series of straight line segments as a bypass path.

What is lost is some precision in the actual location of a bypassing vehicle relative to the obstruction as a function of the actual physical dimensions of the vehicles and obstructions being modeled. This loss of precision is not transferred to the mine encounter calculations as those are strictly a function of the actual vehicle specifications, mine types and mine locations.

The implementation of the model uses the circular assumption to generate a hexagon surrounding the obstruction (see Figure 3-6). The radius of the circle used is the sum of the radius of the obstruction and the radius of the vehicle. A hexagon computed to be tangent to the circular obstruction at the midpoint of each of its six sides is then calculated as the centerline of the vehicle bypass path.

B. PARAMETERS

1. Model Input Parameters

The model allows a wide range of scenarios to be examined easily by allowing most of the model parameters to be changed directly from the input screens. The use of SIMGRAPHICS has made possible an interactive graphic interface for data input. The input interface consists of a series of menus displaying default input values that can be modified, if desired, by the user. A useful byproduct of using SIMGRAPHICS for the input interface is that the allowable ranges of the input values are programmed into the menus. This prevents the entering of input values outside the range for which the model is designed. A list of the modifiable parameters, and their effect, follows.

a. Mine parameters

- Minefield depth. The dimension of the minefield is collected as minefield depth. The depth is important as it impacts directly upon the amount of time required to breach and pass through the minefield. The actual width of the minefield is of lesser importance, since this model operates under the implicit assumption that a breach of a minefield has been decided on as the course of action. For display purposes, the minefield width is calculated to extend for a width equal to the scaled width of the display screen.

- Mine radius. The model treats individual mines as bodies having dimensions (specifically, radius) rather than simplifying the calculations by using a point mass assumption. The user is given the opportunity of entering a specific radius for each type of mine modeled. Each mine is assumed to be a cylinder with a known radius, measured in meters. The model is then able to consider the scenario where the vehicle track encounters only part of the mine.

- Mine depth. The average depth of a mine, measured in meters from the surface down to the top of the mine is an input. In the case of an unburied mine, the depth is zero. The depth of the mine will determine if a plow working at a specific depth actually encounters the mine.

- Reliability of the mine. The probability of a mine detonation given that the mine has been encountered by a vehicle is the reliability of the mine. In reality, this parameter is a complex function of (at least) the fuze type, the amount of time the mine has been in the ground, soil type, and mine design. If a value is not known for this input, the conservative (and default) value is 1.0.

b. Plow parameters

- Plow depth. This value is used to determine if the plow will encounter the mines present in the field (when compared to mine depth). The plow depth will also impact upon the speed with which the plow is able to move through the minefield.

- Plow width. Plow width determines how wide the lane made through the minefield is. The width of the lane will impact on the ability of trailing vehicles to stay in the lane. Like plow depth, this factor impacts upon the speed with which the plow will be able to move through the minefield.

- Plow effectiveness. This is the probability that the plow will remove or neutralize a mine which it physically encounters. Depending upon the size of the mine and the spacing between the plow tines, it is possible for mines that are encountered by the plow to filter through the tines instead of being displaced to the side of the lane. A mine that is not removed or neutralized remains in its original location and with its original reliability.

c. *Vehicle parameters*

- Track width. The actual width of vehicle tracks. This parameter will usually have different values for different types of vehicles. The model will currently handle up to five different vehicle types. The track width is used to determine the pressure footprint of the vehicle.

- Vehicle width. This is the actual width of the vehicle. Again, this parameter will usually differ from vehicle type to vehicle type. This value is used to determine the influence footprint.

- Vehicle length. The actual length of the vehicle. This value is used as part of the calculations for computing the vehicle radius which is used in the circle approximation of the vehicle area.

- Vehicle radius. This is currently not a direct user input value. For the purposes of bypassing obstructions, the model approximates both obstructions and bypassing vehicles as a circle. Since the only types of obstructions currently implemented in the model are disabled vehicles, this value is also used to determine the size of the obstructions. The radius of vehicles and obstructions are used to determine the distance bypassing vehicles must travel in order to get around them. Currently, the radius of a vehicle is a calculated value, found by multiplying the vehicle length by the vehicle width, and then solving for the radius of a circle which has that same area.

- Vehicle speed. This is the value used to determine how fast a vehicle will move through the minefield. The model currently treats vehicle speed as a constant. The plow vehicle's speed is a function of the width of the plow and the depth of the cut it makes in the ground.

- Vehicle navigation accuracy. This parameter is an attempt to determine how closely a trailing vehicle follows in the footprints of previous vehicles, particularly the first (plow) vehicle. Ideally, a trailing vehicle would follow exactly in the footprints of the lead vehicle. However, as was explained in Chapter II, this is rarely the case. When a trailing vehicle deviates from the footprints of the leader, it leaves an area of near zero mine density and enters an area of reduced, full, or increased mine density. This decreases the probability of survival for each trailing vehicle and for the formation as a whole. This parameter is entered as the value of the standard deviation of the x coordinate at each checkpoint in the movement path.

d. *Overwatch parameters*

- Number of overwatchers. The number of red direct fire overwatching systems for the scenario being run. When the model is being run in graphics mode these elements are depicted at the top of the screen, but this location does not

represent their tactical placement. It is only an arbitrary placement for display purposes.

- Type of overwatchers. The model currently plays only one type of overwatching system.

- Rate of fire. This parameter represents the actual rate of fire of the red system as a function of reload and system cycle times. It is input by the user and treated as a constant by the program.

- Acquisition rate. This parameter is actually intended as a combination of detection, acquisition and engagement times for the overwatching systems. The value entered is the mean, β, of a random sample from an exponential distribution.

e. Taskforce parameters

- Number of full width plows. The number of full width plows which will be used to make the breach. The first plow or vehicle has no navigation error, as navigation error is defined as deviation from the pressure or influence footprints of the lead vehicle. Any subsequent plows and vehicles implement the navigation error methodology. Subsequent plows are offset approximately one third vehicle width to either side of the lead plow in order to widen the effective lane through the minefield.

- Number of track width plows. This parameter is the number of track width plows used to make the breach.

- Number of type 1 tracks. This is the number of tracked vehicles of specific type 1. The model will accept values ranging from 1 to 15 for this parameter, but the larger the number the slower the graphics version will run. An additional consideration is that, depending upon the lethality of the scenario and the size of the minefield, the number of disabled vehicles (if excessive) may cause "gridlock" where the size of the resulting obstruction is larger than the boundaries of the minefield. Actually, it is unlikely that any unit commander would willingly put more than a dozen vehicles or so down a single lane of an assault breach. Test cases have shown no difficulty in handling breaching forces of up to 10 vehicles, but no predictions are made for more extreme cases.

- Number of type 2 tracks. The number of tracked vehicles of specific type 2. Just as for the type 1 tracks, the model will accept and attempt to run with up to 15 type 2 tracks. The same cautions apply.

- Number of type 3 tracks. The number of tracked vehicles of a specific type. The same cautions apply.

- Interval between plows and tracks. This parameter allows the user to specify a distance between the breaching vehicles and the trailing assault force. It differs from the interval between vehicles in that it allows separation of the breaching vehicle(s) from the vehicles that are merely trying to get through the minefield. This parameter was implemented, under the assumption that the tactical commander, during a breaching operation, would not want to place combat vehicles in the minefield until a lane had been completed, or at least until all other efforts to provide a lane had been exhausted. If this parameter is set to zero, then the model uses the interval between vehicles for all elements.

- Interval between vehicles. This parameter is used to determine the distance between vehicles moving single file through the minefield. Once set, it remains constant throughout a simulation run.

f. Tables

- Vehicle versus Mine p_k Table. The model allows the vehicle type versus mine type probability of kill table to be edited during the initialization phase. Distinct p_ks may be entered for each vehicle type/mine type combination.

- Red Overwatching Fire versus Vehicle p_k Table. The table of probability of kill given shot table for red overwatching systems versus blue taskforce vehicles can be edited during the initialization portion of a model run.

- Direct Fire Allocation Table. This table allows the user to determine the proportion of direct fire shots that will be allocated to the lead vehicle attempting to negotiate the minefield. Fires not allocated to the lead vehicle are distributed equally to any other vehicles that happen to be in the minefield at the time of the shot.

g. Administrative parameters

- Interval between checkpoints. The interval between navigation checkpoints can be set during the initialization portion of a model run. The value entered impacts on how taskforce vehicle navigation error is played, as the **y** coordinate of each navigation checkpoint is a direct function of the distance between checkpoints. The smaller the interval, the more navigation checkpoints the vehicle will negotiate in the course of moving a specific distance. For each checkpoint, a navigation error in the **x** dimension is calculated.

- Graphics on or off. The graphics option provides a valuable means of validating the model and of displaying the results of a scenario. However, the use of graphics precludes the timely collection of multiple iteration results. To avoid this problem a switch is provided on the input menu which allows the user to turn the graphics off. This action significantly reduces the model run time (by at least an order of magnitude).

- Number of iterations. The model may be set to run multiple iterations with the same input data. For multiple iterations a mean and variance are calculated for the output elements. The number of runs is set during the initialization portion of a model run. If this value is greater than one then the model graphics are automatically turned off.

2. Outputs

Several values are produced as outputs by the model. Additionally, routines exist to capture the values of various data arrays within the model for validation purposes. All input values are recorded as part of the output file, to include the initial seeds used in the random number generators. The output values currently produced are:

- Mine density. This value is expressed in terms of the number of mines present by type per meter of front (linear density) and by the number of mines by type per square meter of minefield area (area density). Linear density is the average number of mines contained in a meter wide strip through the mine field in the direction of formation movement. Area density is the average number of mines per square meter of mined area. Mine density is of critical importance in determining the effectiveness of the mine clearer.

- Mines displaced by plow. This output is expressed as the number of mines displaced by plows during a model run.

- Mine kills by mine type. This output is expressed as the number of mine kills by mine type and by vehicle status. Vehicle status is expressed as either in normal mode or bypass mode.

- Red overwatch shots fired. This value is the total number of shots fired by red direct fire overwatching systems during a scenario run.

- Red overwatch kills. This value is the total number of vehicle kills generated by the red direct fire overwatching systems.

- Time of death. The time of death of any taskforce element is recorded and available as an output.

- Time of completion. For each vehicle surviving the breaching effort, the time it completed the traversing the lane is recorded and available as an output.

C. THE MODEL

Figure 4-1 is a flow diagram of the encounter event logic used in the model.

1. Events

Two types of events occur within the model. The first type, hereafter called "encounter" events, are composed of those events which take place as a result of vehicle movement. The second type of event, hereafter labeled "fire" events, are made up of those events which take place regardless of vehicle movement.

a. Encounter events

There are three types of encounter events. They are **navigation checkpoint encounters**, henceforth called cp encounters, **mine encounters**, and **obstruction encounters**.

There are four varieties of cp encounters. These are initial checkpoint encounters, movement checkpoint encounters, bypass checkpoint encounters, and final checkpoint encounters. CP encounters are used to monitor and control the navigation of the taskforce through the minefield. All taskforce vehicles start at their respective initial checkpoints. The time of arrival at an initial checkpoint is determined by the vehicles place in the column, vehicle speed and the interval between vehicles. Movement checkpoints occur at user specified intervals and when connected by line equations, make up each vehicles movement path. At each movement checkpoint the path equation used to determine the exact location of the vehicle changes. Bypass checkpoints serve a similar function, however they are calculated as needed and then spliced onto the original movement path. Upon reaching the final checkpoint the vehicle has successfully negotiated the minefield.

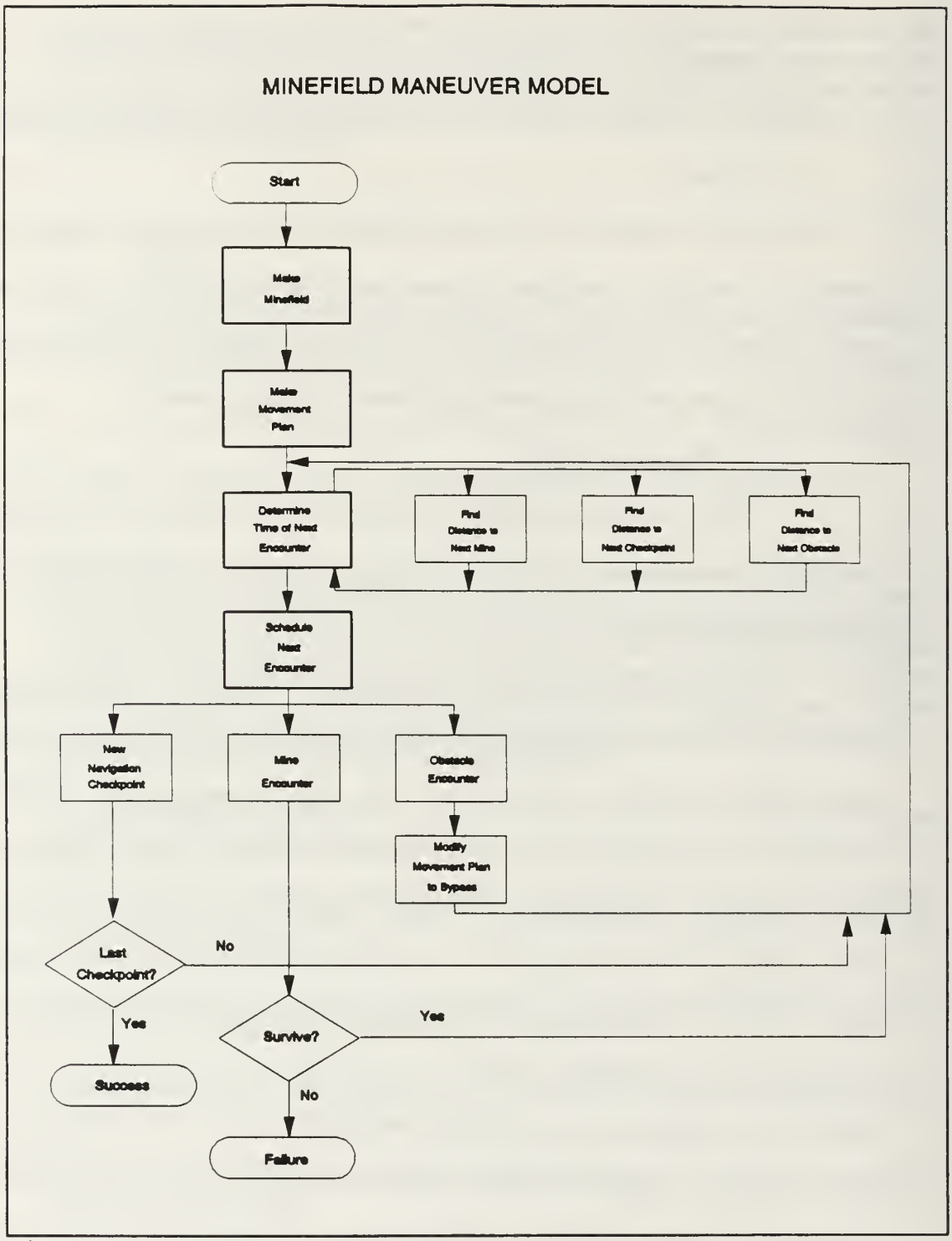
MINEFIELD MANEUVER MODEL
Start
Make Minefield
Make Movement Plan
Determine Time of Next Encounter
Find Distance to Next Mine
Find Distance to Next Checkpoint
Find Distance to Next Obstacle
Schedule Next Encounter
New Navigation Checkpoint
Mine Encounter
Obstacle Encounter
Modify Movement Plan to Bypass
Last Checkpoint?
No
Yes
Survive?
Yes
No
Success
Failure

Figure 4-1

Mine encounters occur when the movement or bypass path and the physical dimensions of a vehicle are combined with the location, size and fuzing mechanism of a mine resulting in an engagement.

Obstruction encounters can occur only after at least one vehicle has been disabled, with at least one vehicle trailing it (the disabled vehicle). An obstruction encounter initiates the generation of a bypass map consisting of the bypass checkpoints necessary for the trailing vehicle to bypass the obstruction.

Each vehicle has a maximum of one encounter event on the event calendar at any instant in time. Disabled vehicles no longer generate encounter events. When an encounter event is taken off of the calendar and completed, a new encounter event for that vehicle, if appropriate, is determined and scheduled.

This is done by a routine named **NEXT.ENCOUNTER** which performs a function call to each of three different routines, named **DISTANCE.TO.CP**, **DISTANCE.TO.MINE**, and **DISTANCE.TO.OBS**. Each of these routines determines the distance between the calling vehicle and its next encounter of each respective type. The routines then return the distance, and if appropriate, identification of the encountered entity (mine ID, or obstruction ID) to the **NEXT.ENCOUNTER** routine. **NEXT.ENCOUNTER** then compares the distances and determines which event is most eminent. A call is then made to the routine **DELTA.TIME** which computes and returns the time it will take that vehicle, moving at its constant speed, to move the appropriate distance. **NEXT.ENCOUNTER** then schedules the next event for that vehicle, either a **NEW.CP**, **MINE.ENCOUNTER**, or **OBSTACLE.ENCOUNTER**, at that time in the future.

This process will continue until all taskforce members have been disabled or have reached their final navigation checkpoints. The only encounter event which can disable a vehicle is a mine encounter. If a vehicle is disabled during a mine encounter, a routine called **CALENDAR.UPDATE** is used to remove all encounter events from the event calendar. This is done to prevent events from occurring which may no longer be valid. After all encounter events have been removed the routine causes each active element to redetermine and reschedule its next event.

b. Fire events

There are two types of fire events. **Direct fire** events and **indirect fire** events. Direct fire events are those events that take place in a simulated line of sight mode where the firer is assumed to be able to see the target being fired upon. Each red direct fire overwatch system will have only one fire event on the event list at any time. Direct fire events will not be directed at a specific vehicle until the event is removed from the event list. At that time the routine **DIRECT.FIRE** determines the actual target using the fire allocation algorithm previously discussed and resolves the engagement. There may be multiple direct fire events apportioned to the same target vehicle, but only one per firing vehicle at any one time.

Indirect fire events refer to the use of simulated artillery, where the firer is actually attempting to hit a spot on the ground. Having a line of sight to the target is not required or expected. Multiple indirect fire events may be on the event list, but they are directed at impact points and not specific vehicle entities.

Should a vehicle be disabled by direct or indirect fire, **CALENDAR.UPDATE** is again called in order to clear the calendar of encounter events and reschedule encounter events for all active vehicles.

V. EXAMPLE PROBLEM

This chapter documents a demonstration of the model through the use of an example problem. One of the issues that has arisen during the design of the new Combat Mobility Vehicle concerns the choice of an appropriate width for the blade attached to the front of the vehicle. The blade may be used for a variety of purposes, including as a plow for the breaching of minefields.

As discussed in Chapter II, the utility of a plow blade is dependent on tradeoffs made between the effectiveness of the lane created by its use and the survivability of the breaching vehicle. Intuitively, the wider the blade, the more dirt the vehicle must push, and the slower the vehicle will be able to move. If the lane is not wide enough to be safely negotiated by the trailing vehicles, then one or more of them may become disabled by a mine, thereby blocking the lane. Finally, the longer any vehicle remains exposed to overwatching fires while in the minefield, the more likely the event of its being disabled.

To demonstrate the potential utility of this model with regard to examining the effects of parameter changes on the scenario outcome, the following test case is presented.

A. SCENARIO

The purpose of this scenario will be to evaluate the impact of mine plow width on the number of casualties inflicted upon a taskforce conducting the breach of a

minefield. There are many variables which will impact on the results of a minefield breaching operation. The ones included in this scenario are:

- the density, depth and makeup of the minefield,
- the volume and accuracy of overwatching direct fires,
- the width of the cleared lane made,
- the speed of the breacher,
- and the navigation accuracy of the trailing taskforce.

1. The Minefield

The scenario minefield is a surface laid, scattered minefield. Scattered refers to the fact that the mines are placed randomly, using a uniform distribution, within a minefield of 50 meters depth and 150 meters width. Three types of mines will be modeled in the makeup of the field. Table 5-1 shows the type, radius and probability of kill parameters used for the mines in this scenario.

TABLE 5-1 MINE DATA

Mine Type	Linear Mine Density	Radius of Mine	P_k vs Full Plow	P_k vs Track
Pressure	.2	.1 m	.05	.9
Influence	.1	.1 m	.05	.9
Contact	.025	.1 m	.5	.9

2. The Taskforce

The attacking force will consist of one full width blade breacher and five tracked vehicles for a total of six vehicles. A baseline case will also be run in which a taskforce consisting of six tracked vehicles with no plow will attempt to negotiate the minefield. Table 5-2 displays the model input parameter characteristics of the breaching force.

TABLE 5-2 VEHICLE DATA

	Width	Length	Track Width	Nav. Error	Speed
Plow	3.48m	9.03 m	.635 m	none	varied
Track	3.48m	9.03 m	.635 m	.5 m	10 kph

3. Plowing

The actual amount and type of force used by a prime mover to push a plow is a complex function of soil type, plow design, and plowing depth. The forces involved can be divided into two categories; the system tractive forces, and the plow draft forces. These forces are defined as follows.

> System tractive force - The amount of force exerted by a prime mover against a given soil that can be used for maneuverability/mobility (turning, climbing, accelerating, etc.). This includes the amount of force the soil is able to generate as resistance versus the amount of force generated by the prime mover.
>
> Plow draft force - The amount of force applied to plow tines, skids, and moldboards to conduct plowing operations in a given soil. [Ref 5. p IV-3]

Simplifying the problem of plow draft force considerably, we make the statement that under a given set of conditions, the faster the plow is pushed, the greater the draft force required. The relationship between plow draft force and prime mover velocity is roughly linear.

Continuing to simplify, we make the statement that with regard to tractive force, under a given set of conditions, the faster the plow is pushed the less tractive force is available. The relationship between tractive force and velocity is not linear.

The draft force required to move the plow is subtracted from the tractive force available to the prime mover. A positive net tractive force is necessary if the prime

mover is to move forward. This means that no matter how much horsepower the prime mover has available, there is a point where the ambient conditions will not permit an increase in the forward velocity of the plow. The prime mover will be simply "spinning it's wheels".

Figure 5-1 illustrates the relationship between velocity and both draft and tractive forces. Figure 5-1 was derived from data extracted from a study done on the engineering development of a mine clearing plow system [Ref 8. p.IV-14]. The intersections between the tractive force and draft force lines are the points where further increases in velocity are not possible. The abbreviation TWMP stands for track width mine plow and FWMP stands for full width mine plow.

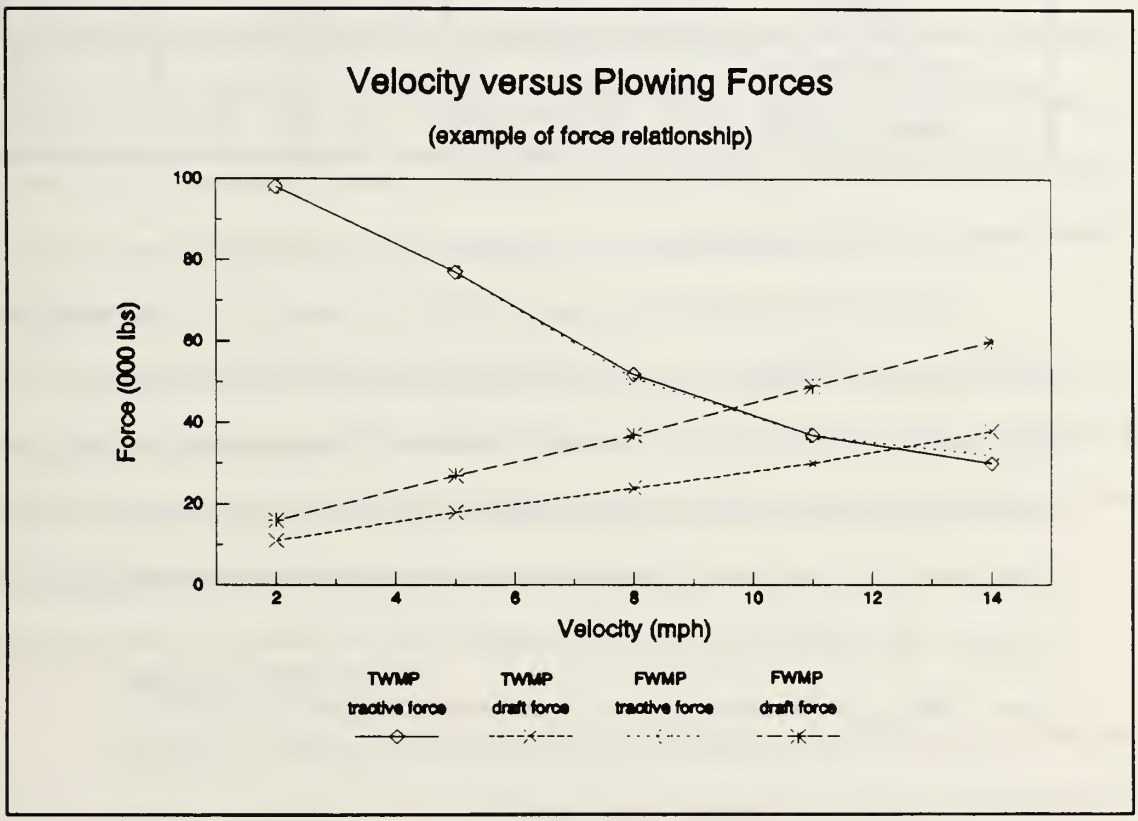

Figure 5-1

It is apparent that any specified prime mover will, under stipulated soil conditions, be able to move a designated plow at a certain maximum velocity. This point will be the intersection of the draft forces and the tractive forces generated under the aforementioned soil conditions. The calculations and computer programming required to calculate this point under any reasonable variety of soil conditions and vehicle configurations are of sufficient magnitude to be the subject of several volumes. For this example, we will simplify the situation by using the values in Table 5-3 for the maximum velocity of our plowing system under the given plow width.

TABLE 5-3 PLOW DATA

System	1	2	3	4	5
Plow Width (m)	4.5	5.0	5.5	6.0	6.5
Velocity (kph)	8.0	6.0	4.6	3.8	2.8

4. Red Overwatch

The red force overwatching the minefield will consist of a single red direct fire system which will have a rate of fire of one round every 30 seconds, and a mean acquisition rate of one target every 40 seconds. The overwatching system will not begin attempting to acquire a target until the system is prepared to fire. This means that rate of fire and time to acquire are treated as separate, non-overlapping periods of time. The time period between successive shots is the sum of these two periods of time. The p_k given a shot will be .20 against both plows and tracks.

B. TEST PLAN

The goal is to examine the number of taskforce casualties resulting from a breaching attempt as a function of plow width. The minefield model will address this goal by holding most parameters constant, varying only the plow width and the associated speed of the plowing vehicle. The interval between the plow and the trailing vehicles will be set for 300 meters, a value sufficient to prevent taskforce vehicle collisions under the described scenario.

The mean number of casualties for each plow configuration will be determined for 120 iterations of the model. A baseline scenario will be run with no plow system in order to determine the benefit derived from having a plowing system. Vehicle casualties will be subdivided into mean casualties caused by mines, mean casualties caused by direct fires, and mean casualties caused by mines while bypassing obstructions (a subset of mean casualties caused by mines). The mean values will then be examined to determine if differences are apparent and to see if the model results make sense.

The distribution of casualty counts will be determined over each group of 120 iterations. The number of casualties by vehicle position in the taskforce column will also be collected and presented. Finally, the survival rate of the different plow configurations will be determined and displayed.

C. PROBLEM RESULTS

Figure 5-2, presents the mean casualties inflicted during 120 iterations for each of the system configurations. The trend lines support the original reasoning with regard to the consequences of plow velocity and lane width. Note that the "no plow"

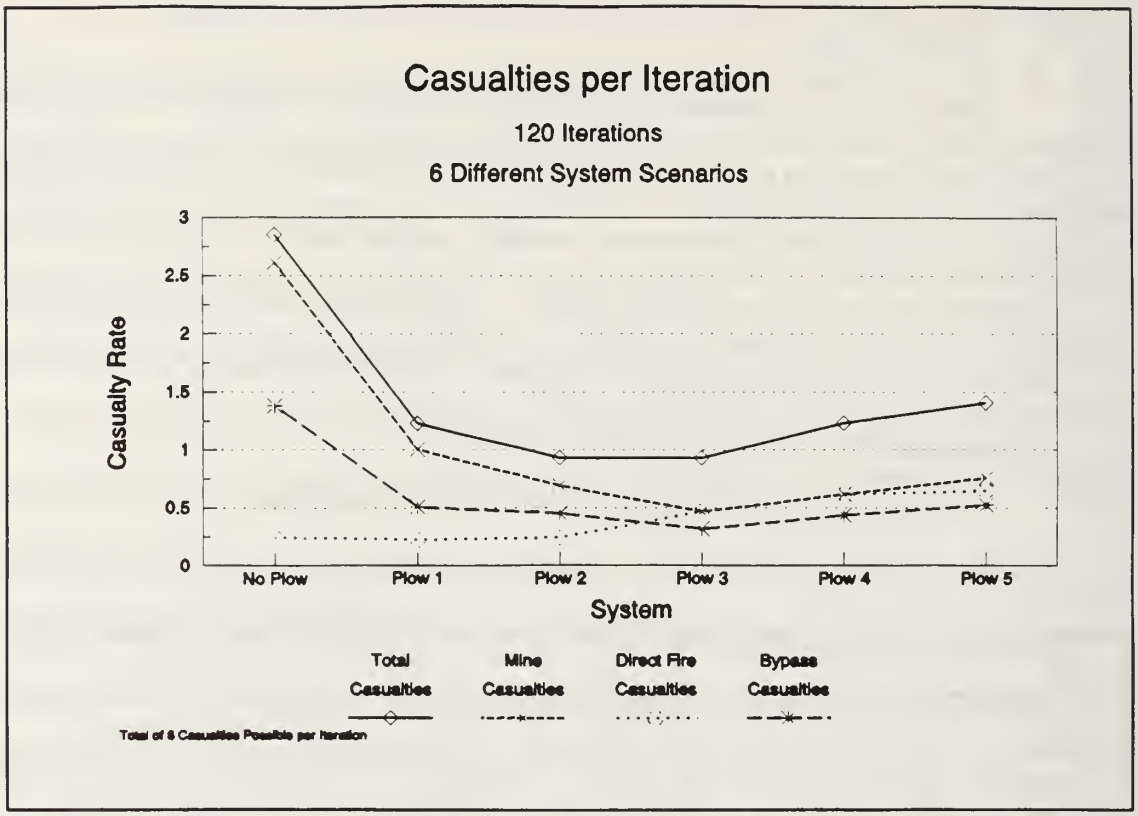

Figure 5-2

scenario results in the most mine inflicted casualties and the most total casualties. Casualties from direct fire go up with the decrease in breacher speed (increase in plow width). As plow width increases, the number of mine casualties initially goes down, but as direct fire casualties rise, so do the mine casualties, particularly those mine casualties inflicted during the conduct of obstruction bypassing. Another cause for the rise in mine casualties may be the destruction of the plow by direct fire prior to the completion of the breach.

The results indicate that under the given conditions, either Plow 2 (5.0 meters wide) or Plow 3 (5.5 meters wide) would result in the fewest overall taskforce casualties.

Figure 5-3 depicts the distribution of taskforce casualties results over the 120 iterations. The values generated do not contradict the expected results. The results also indicate that any plow system is better than attempting to negotiate the minefield with no plow.

Model Run Results

120 Iterations

Run Casualty Counts

	No Plow	Plow 1	Plow 2	Plow 3	Plow 4	Plow 5
No Casualties	16	53	76	66	50	51
1 Casualty	17	32	17	27	39	34
2 Casualties	22	9	11	12	14	8
3 Casualties	19	14	5	5	4	8
4 Casualties	18	7	2	5	2	7
5 Casualties	16	2	4	4	5	5
6 Casualties	12	3	5	1	6	7
Total Casualties	342	148	112	112	148	169

Figure 5-3

Figure 5-4 displays the number of iterations resulting in vehicle casualties at specific vehicle positions in the taskforce column. If present, the breacher will always be the first vehicle in the column. As expected, the slower the breacher moves, the greater the chance it will become a casualty.

Model Run Results

120 Iterations

Number of Iterations where Specific Vehicle became Casualty

	No Plow	Plow 1	Plow 2	Plow 3	Plow 4	Plow 5
1st Vehicle	72	25	25	35	39	49
2nd Vehicle	70	29	18	23	34	28
3rd Vehicle	62	24	15	15	21	27
4th Vehicle	59	30	22	12	19	27
5th Vehicle	41	18	17	14	13	20
6th Vehicle	36	22	15	13	22	18

If a Plow is Present it will be Vehicle 1

Figure 5-4

Figure 5-5 emphasizes this point by showing the actual survival rates for the five different plow configurations as a result of vehicle velocity, which equates to the amount of time spent exposed in the minefield.

Actual studies to determine the appropriate plow width would entail examination of a much greater selection of soil types, minefield configurations, and overwatching fire conditions. Also, this example uses fairly small sample sizes. Any study intended for actual design decisions should include a larger number of scenario iterations.

This example demonstrates the potential utility of the methodologies and the model for examining the problems associated with minefield breaching operations. The model may be particularly useful for examining complex problems that have no

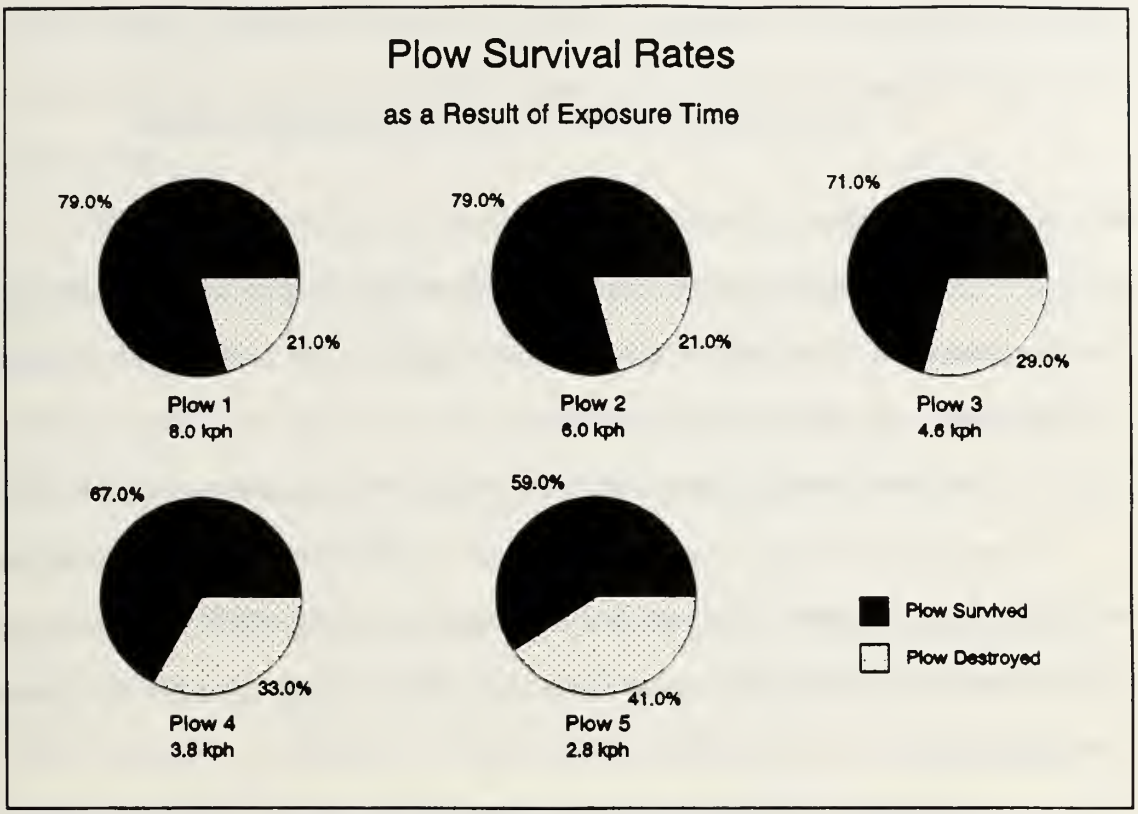

Figure 5-5

closed form solution and for which the costs and hazards of actual testing are prohibitive.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. RESULTS

The methodologies and algorithms presented in this model provide a good foundation for the effective high resolution, high fidelity, modeling of the dynamics involved in the traversal of minefields.

The model is highly portable. The program was written and tested on an IBM PC compatible computer, and has been run on both the 80286 and 80386 systems. The primary portability limitation is the availability of the SIMSCRIPT programming environment. Further details concerning system requirements are provided in Appendix A.

The model is entirely menu driven, allowing the user to change any of the input parameters without editing a file. The range of allowable input values is controlled by the SIMGRAPHICS input forms. The model outputs are automatically written to a data file for study and analysis.

The model is written in SIMSCRIPT and includes the icons and coding to run using SIMGRAPHICS. The graphics are a user option that is turned off should the user decide to run multiple iterations in a single session. The use of SIMGRAPHICS provides a visual validation of the model and of the model results. Users will have added confidence of being able to see how the model arrives at its results.

An unfortunate limitation of the SIMGRAPHICS feature is that it is specific to the machine environment in which it was developed. This means that the graphics in this model will only work on a DOS system based machine. A list of the icons as

well as a figure depicting each one is provided in Appendix D to allow users, interested in transporting the model to a different system environment, a template for the icon construction.

The resulting model provides a basic high resolution modeling capability for the study of minefield dynamics with potential utility in either countermobility or mobility studies. The model can be used for both offensive and defensive scenarios and has a built in capability to examine a wide range of "what if" scenarios. The model also has potential as an optimization tool to examine such issues as optimum plow widths and optimum mine densities/mixes.

The use of the model as a low level, high resolution modeling tool, allows the user to collect a wide range of minefield breaching data, providing the military modeler with a capability to "validate" the minefield breaching portions of higher level, lower resolution military models. For example, as a high resolution model, the program can be used to determine expected unit delays and vehicle attrition as the result of an encounter with a specific type of minefield. These outcomes can then be used to evaluate the performance of higher level models to determine if the delays and casualty assessments they impose are reasonable, or the results can be used as data for the higher level models.

B. FUTURE ENHANCEMENTS

This model has been designed to allow easy modification of many of the model inputs. In addition, basic tools and functions are provided that allow for the easy addition of additional minefield objects. Possible improvements include adding

additional breaching options such as mine rollers, line charges, and flails. Additional minefield features might include patterned minefields, multiple lanes, blue overwatching fires, antitank ditches, obscurants, and some of the more modern mine types. The addition of indirect fires to the overwatching fires as well as the ability to modify the rate of overwatching fires as a function of time should also be considered as a future enhancement.

LIST OF REFERENCES

1. The United States Army Posture Statement FY 92/93, Trained and Ready, by M.P. Stone and C.E. Vuono, February 1991.

2. Department of the Army, Review of Army Analysis, Department of the Army Special Study Group, 1979.

3. Engineer Studies Center Report USAESC-R-86-6, The Engineer Model Improvement Program Plan", US Army Corps of Engineers, Fort Belvoir, VA, 1988.

4. Garland, M. W., KHAFJI: A COMBAT SIMULATION, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1991.

5. McLean Research Center, Inc., Draft Technical Report, System/Subsystem Integration Analysis in Support of Full Scale Engineering Development of the Mine Clearing Plow System for the Main Battle Tank., by Chitwood, Page, Schilling, Tittsworth, August 1984.

BIBLIOGRAPHY

Russell, Edward C. Building Simulation Models with SIMSCRIPT II.5 ®. CACI Products Company, La Jolla, CA., October 1989.

CACI Products Company, SIMSCRIPT II.5 ® Reference Handbook. Third Edition, CACI Products Company, La Jolla, CA., January 1989.

Kiviat, P. J., et al. SIMSCRIPT II.5 ® Programming Language.
CACI Products Company, La Jolla, CA., December 1988.

Law, Averill M., and Christopher S. Larmey, An Introduction to Simulation Using SIMSCRIPT II.5®., CACI Products Company, La Jolla, CA., 1984.

Robert R. Wallace, et al COUNTERMINE SYSTEMS STUDY: PART 1A: BASELINE SYSTEM DESCRIPTION, Army Mobility Equipment Research and Development Center, Sept 1972.

General Dynamics, MINE CLEARING PLOW SYSTEM, final technical report, Dec 1985.

Wallace, Young and Felts, Countermine Systems Study: Part IA: Baseline System Description, U.S. Army Mobility Equipment Research and Development Center, September 1972.

Department of the Army, FM 17-95, Cavalry Operations, 14 February 1986.

Department of the Army, FM 20-32, Mine / Countermine Operations, December 1985.

APPENDIX A - SYSTEM REQUIREMENTS

The model is written in SIMSCRIPT II.5®, a free-form, English-like, general purpose simulation programming language. SIMSCRIPT II.5 is a product of the CACI Products Company. The model also makes use of SIMGRAPHICS, a companion product which allows programs written in SIMSCRIPT II.5 to include animated interactive graphics.

Running the model requires SIMSCRIPT II.5 to be installed on the machine. Running the SIMSCRIPT compiler on a PC requires a math coprocessor and at least 640K of memory. The graphics have been run on both an EGA and a VGA monitor with no difficulty.

The model was developed on an IBM clone 286 machine, running at 16 MHZ, with a math coprocessor, and VGA graphics. The source code is claimed by CACI to be portable over a wide range of systems with only minor, system specific modifications required.

The graphics, unfortunately, will not transfer to a non-DOS environment. Should the user desire to operate the program on some other type of system which supports graphics, a SUN workstation for example, it will be necessary to recreate the graphic images (icons) and input displays (forms) stored as part of the program using the system specific SIMGRAPHICS software. A list of icons used and figure for each is included in Appendix D. Additionally, paper copies of the input forms are provided.

APPENDIX B - RUNNING THE MODEL

This appendix contains instuctions for running the model and copies of the SIMGRAPHICS forms used to modify the model inputs. The input forms are included because although SIMSCRIPT code will run on a variety of computer systems, the SIMGRAPHICS portion of the code is unique to the type of computer system the graphics were generated on. To allow users the ability to recreate the forms on their specific platforms, they are reproduced in hard copy here. The character strings contained in brackets "< >" are the field identifications used by the SIMSCRIPT program to read the inputs.

Figure B-1 shows an overview of how the various model menus are interconnected. The first menu that will come up upon running the program is the master menu depicted in Figure B-2. From the master menu all other program menus can be accessed. Upon completion of input modifications, the user will return to the master menu and activate the start button.

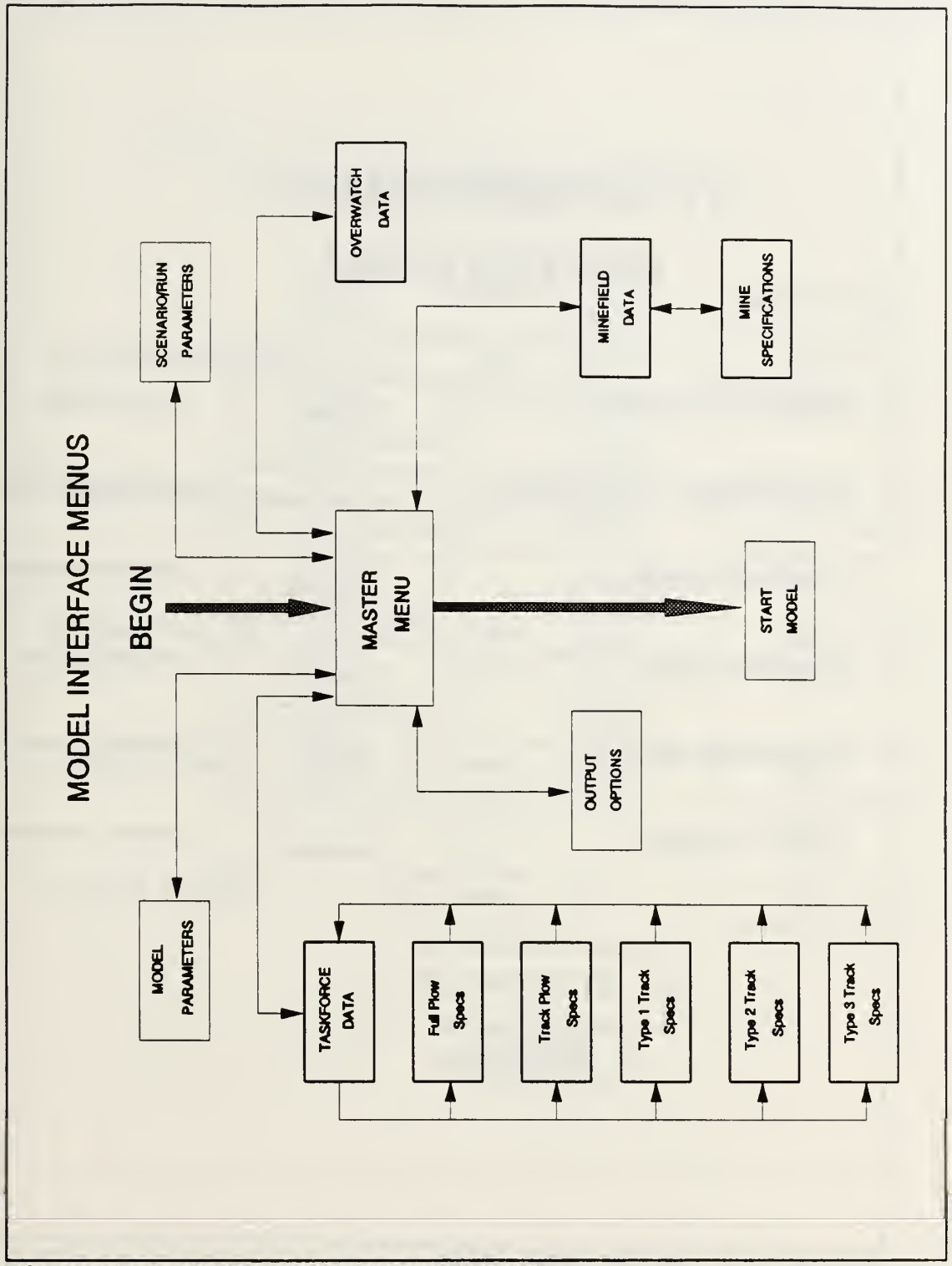

Figure B-1 INPUT FORM FLOW DIAGRAM

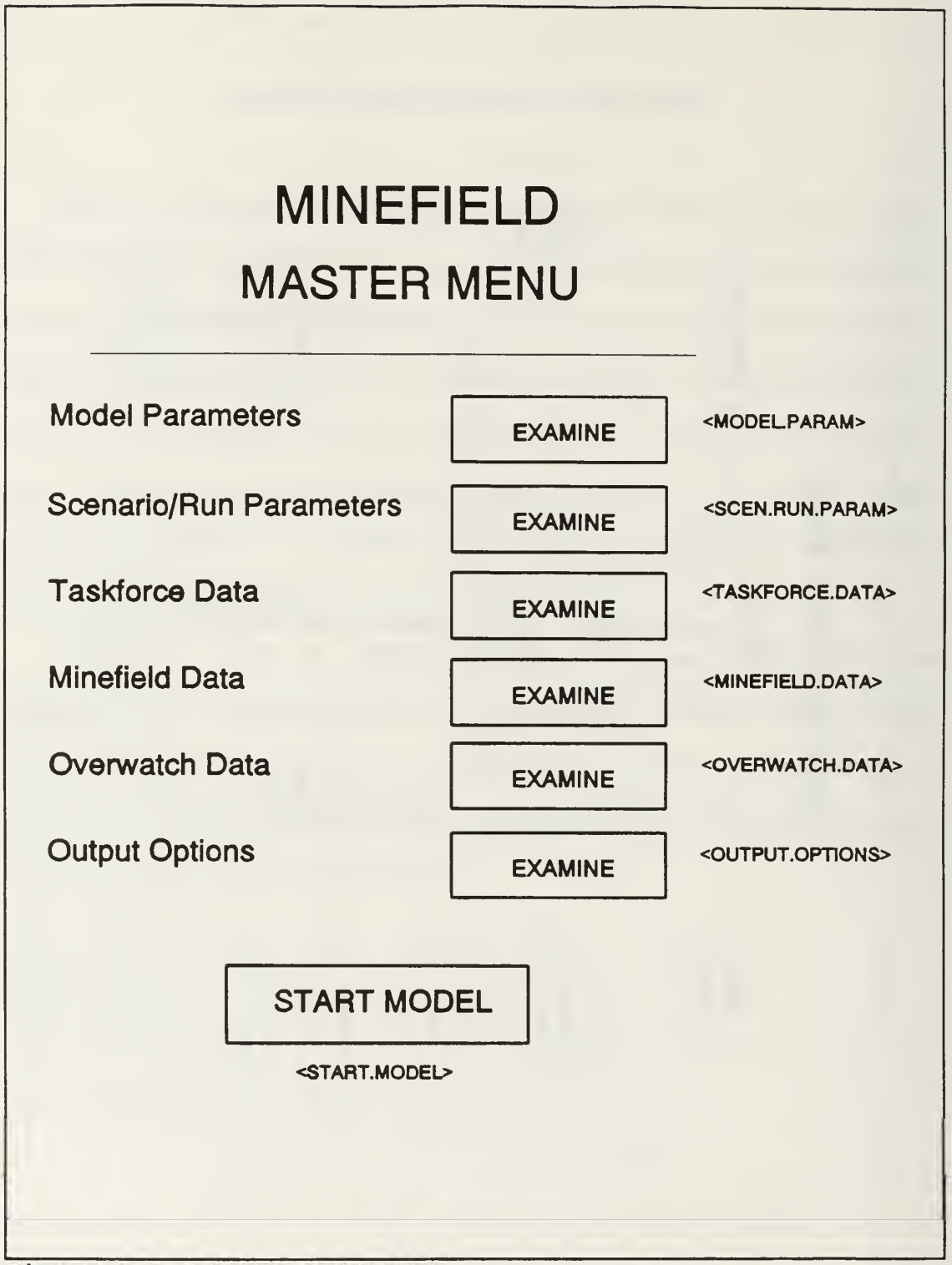

Figure B-2 MASTER MENU FORM

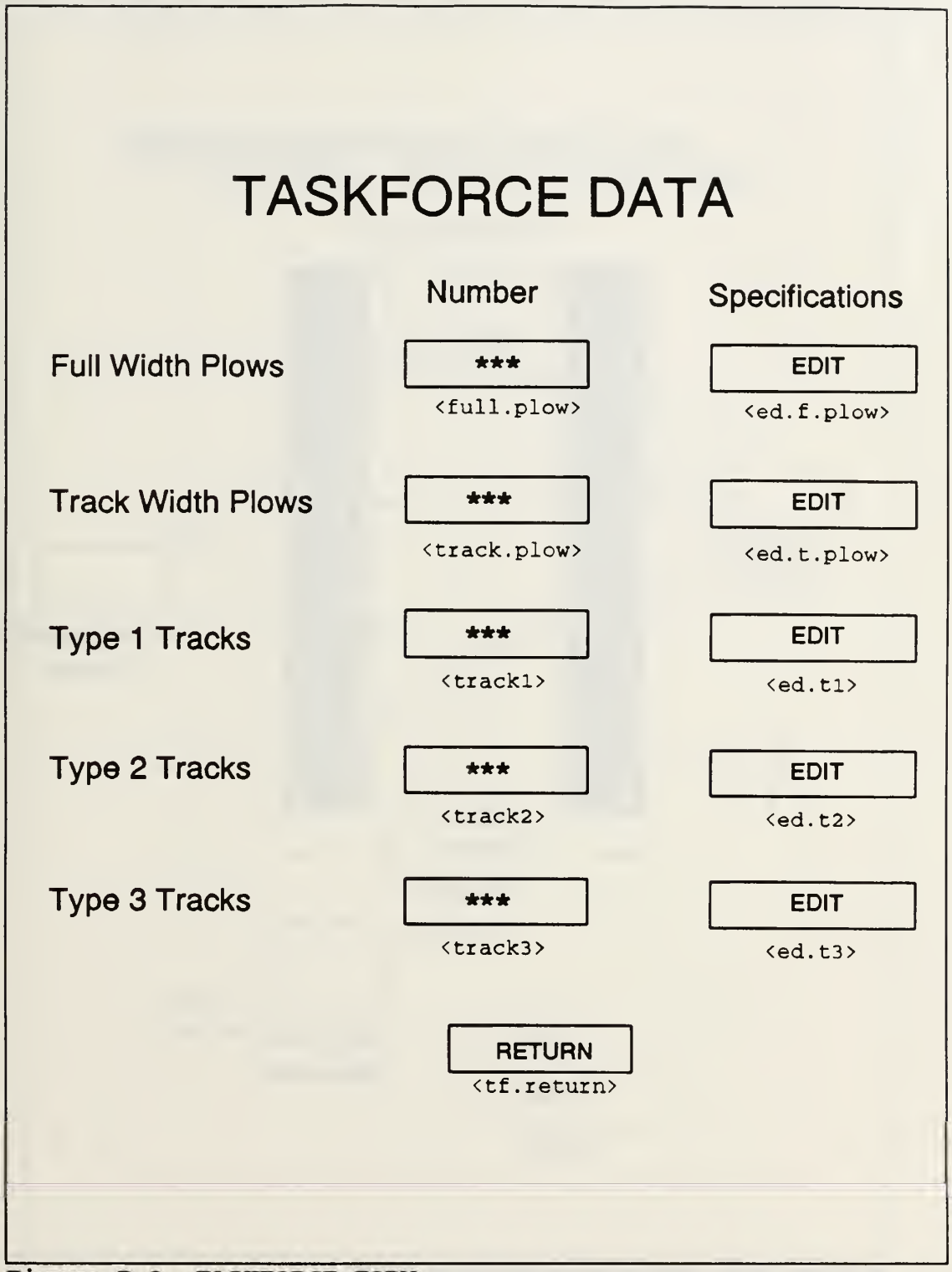

Figure B-3 TASKFORCE FORM

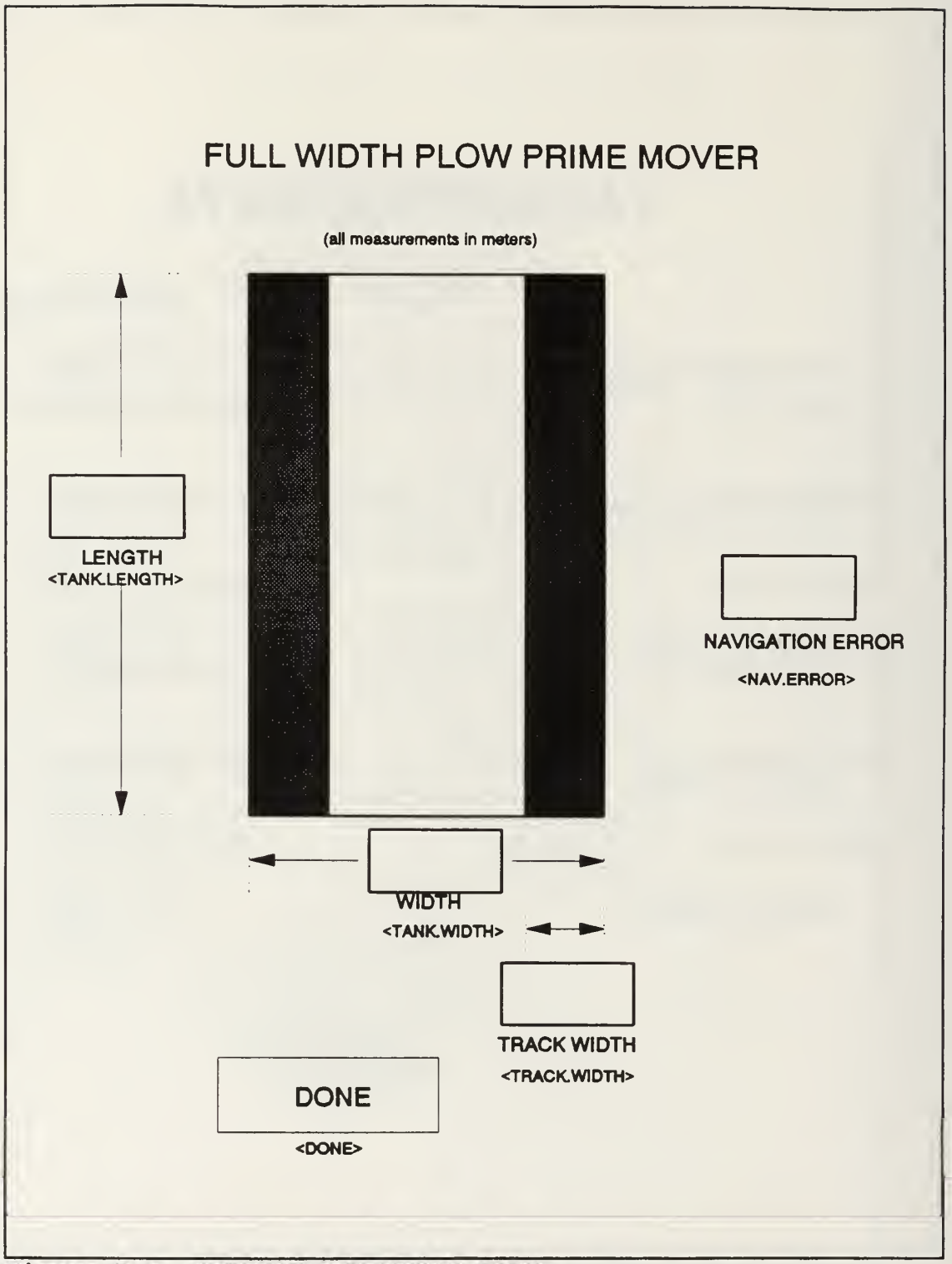

Figure B-4 VEHICLE DIMENSION FORM

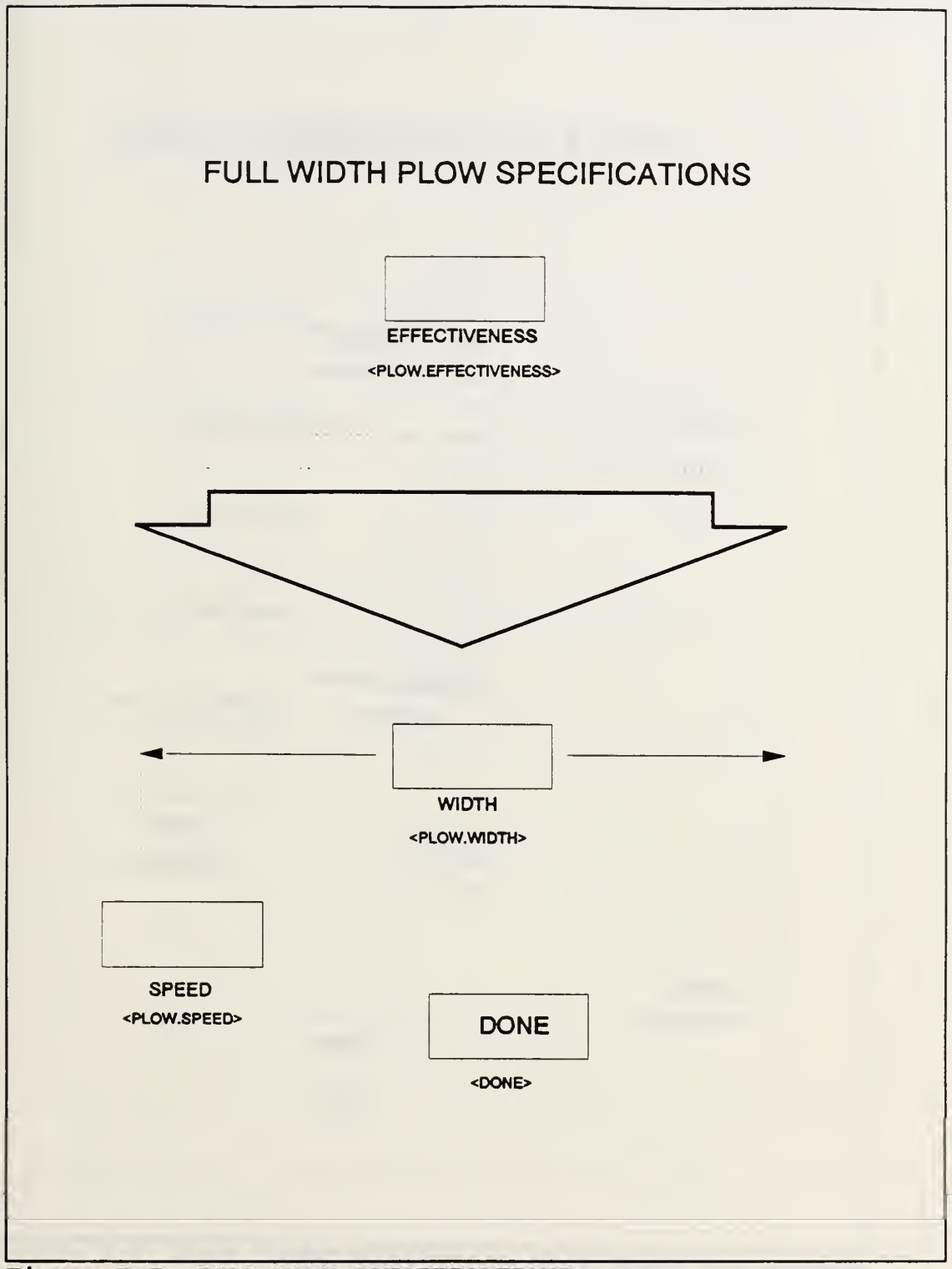

Figure B-5 FULL PLOW SPECIFICATIONS

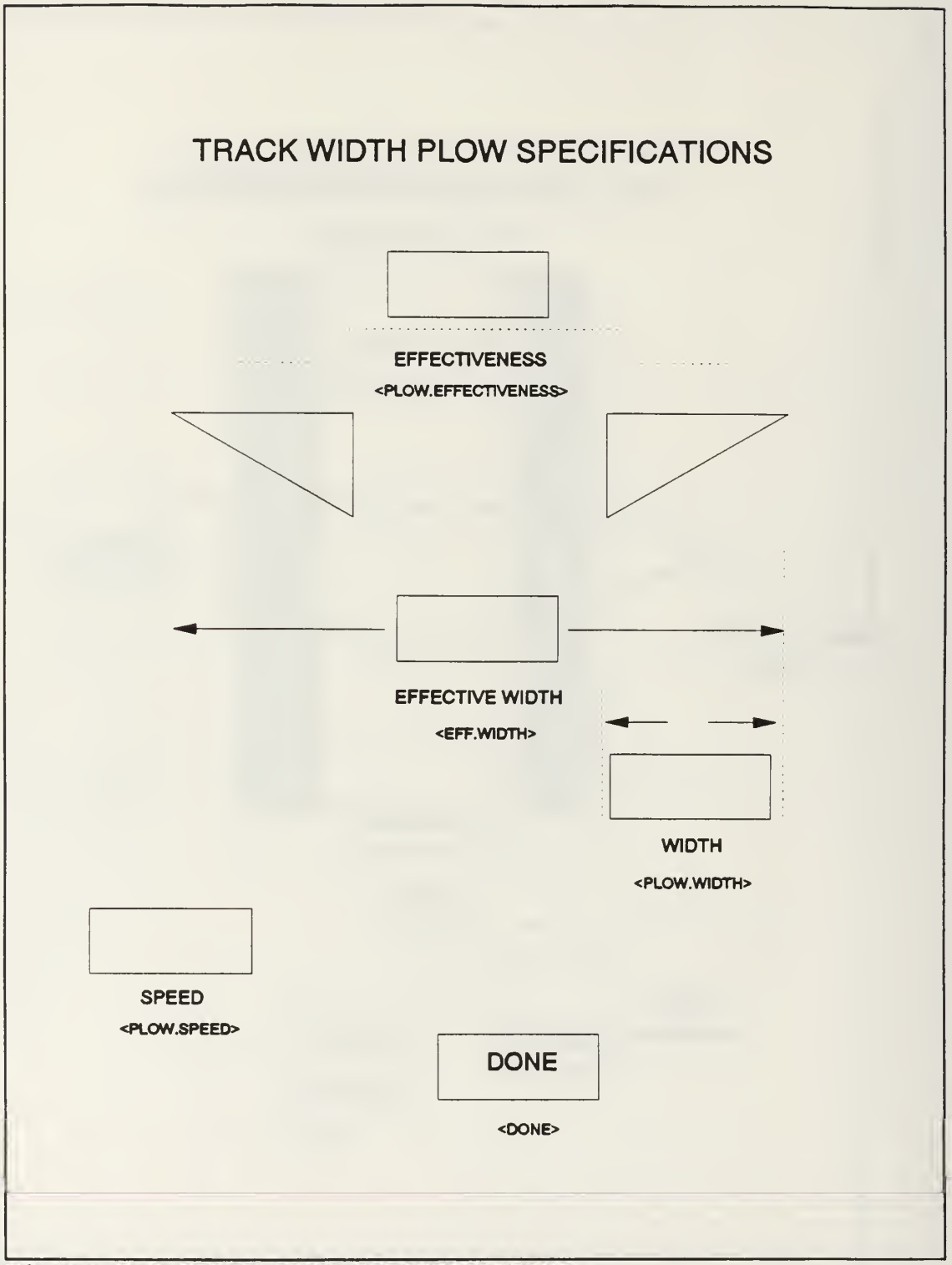

Figure B-6 TRACK PLOW SPECIFICATIONS

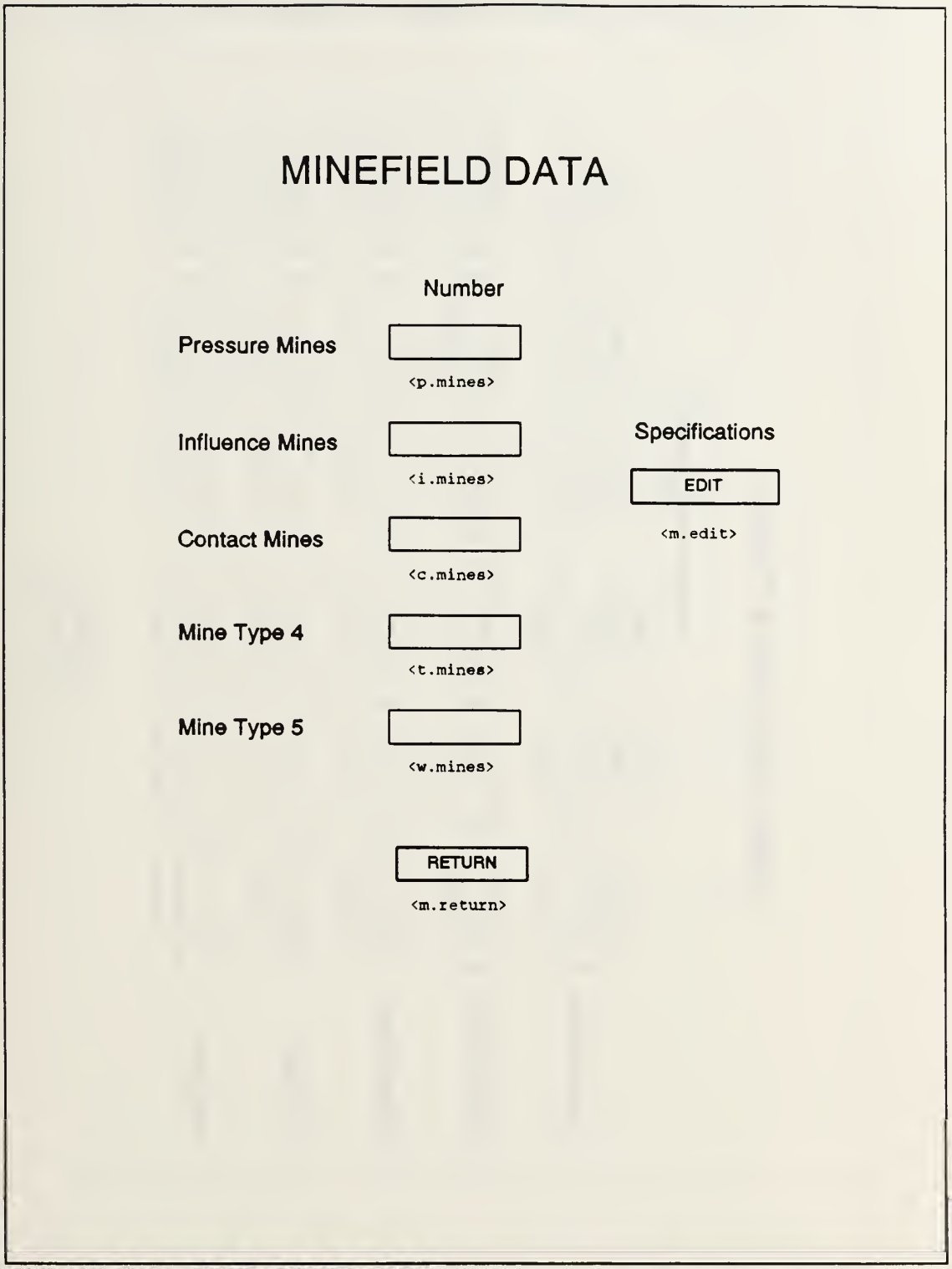

Figure B-7 MINE INPUT FORM

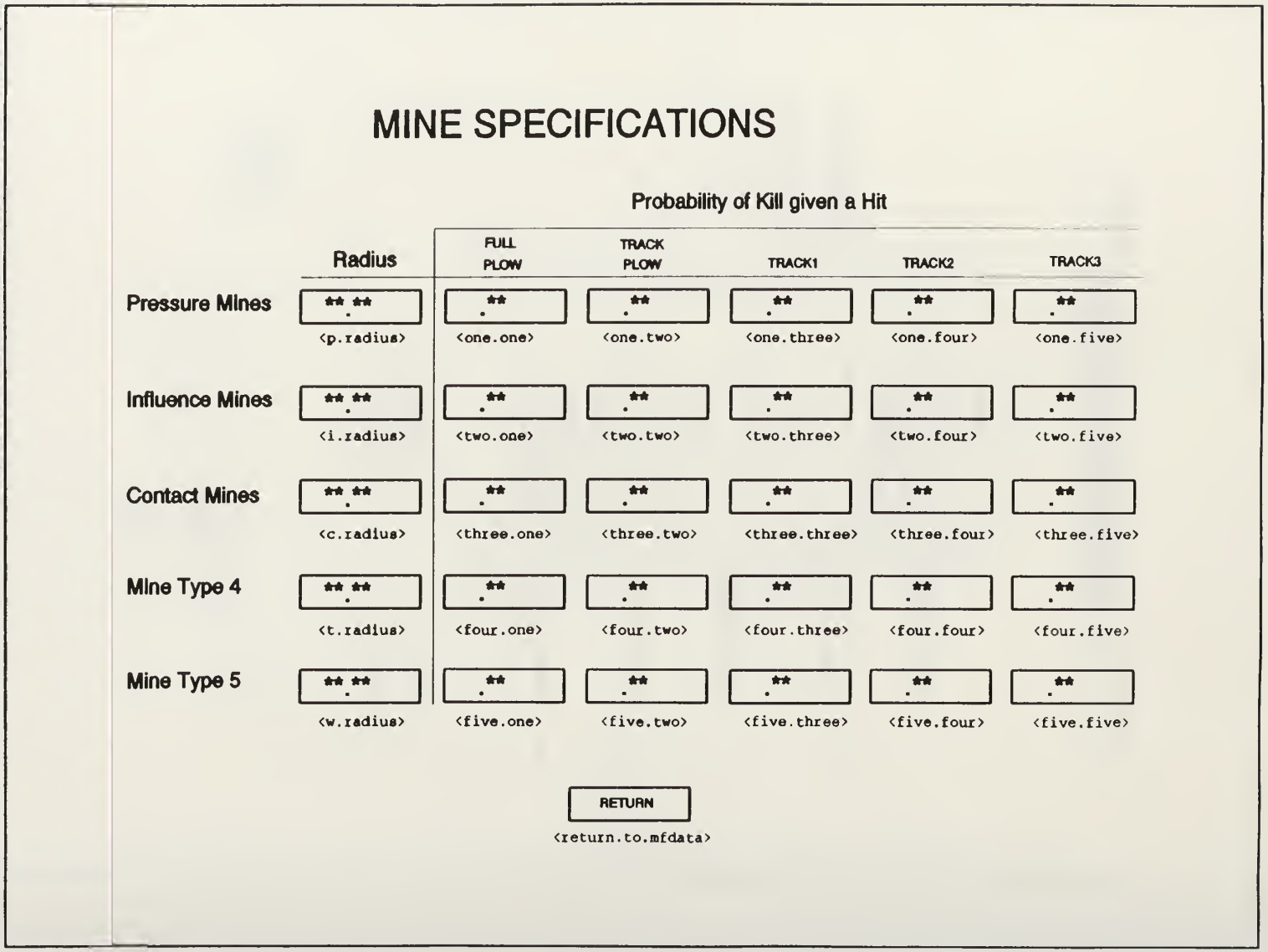

Figure B-8 MINE SPECIFICATIONS

APPENDIX C - SOURCE CODE LISTINGS

```
 1  PREAMBLE
 3  normally mode is real
 4
 5  EVENT NOTICES include GRAPHIC.UPDATE, STOP.SIM
 6    every START has an ID
 7    every MINE.ENCOUNTER has an ID, a MINE.ID
 8    every NEW.CP has an ID
 9    every OBSTACLE.ENCOUNTER has an ID
10    every DIRECT.FIRE has an ID
11
12  TEMPORARY ENTITIES
14    every ELEMENT has a ELEMENT.X, an ELEMENT.Y,
15      a STATUS, an ELEMENT.TYPE, a SPEED, a WIDTH,
16      a LENGTH, a TRACK.WIDTH, an ELEM.RADIUS, a PLOW.EFFECT,
17      a PLOW.WIDTH, a P.STATUS and an ELEMENT.NUM
18
19      define ELEMENT.X and ELEMENT.Y as real variables   ''element location
20      define SPEED as a real variable
21
22                          ''status is 0 if inactive, 1 if normal movement
23                          ''2 if bypassing
24      define STATUS as an integer variable
25
26                          ''type is 1 if full plow, 2 if track plow, 3,4,5
27                          ''if some other type tracked vehicle
28      define ELEMENT.TYPE as an integer variable
29
30                          ''radius of element defined as the radius
31                          ''of the circle with same area as vehicle
32      define ELEM.RADIUS as a real variable
33      define WIDTH, and LENGTH as real variables
34      define TRACK.WIDTH as a real variable
35      define ELEMENT.NUM as an integer variable
36
37                          ''plow effectiveness, plow width and plow status
38                          ''has no effect unless vehicle is of type 1
39                          ''or type 2
40      define PLOW.EFFECT and PLOW.WIDTH as real variables
41      define P.STATUS as an integer variable
42
43    every OBSTACLE has an OBSTACLE.X, an OBSTACLE.Y,
44      a PT1.X, a PT1.Y, a PT2.X, a PT2.Y, a PT3.X, a PT3.Y,
45      a PT4.X, a PT4.Y, a PT5.X, a PT5.Y, a PT6.X, a PT6.Y,
46      a SLOPE.A, a SLOPE.F, a SLOPE.C, a SLOPE.D,
47      a Y.INT.A, a Y.INT.F, a Y.INT.C, a Y.INT.D,
48      and an OBS.RADIUS
49
50      define OBSTACLE.X and OBSTACLE.Y as real variables ''obstacle center
51      define OBS.RADIUS as a real variable        ''obstacle radius
52      define PT1.X and PT1.Y as real variables    ''hex point 1 coordinates
53      define PT2.X and PT2.Y as real variables    ''hex point 2 coordinates
54      define PT3.X and PT3.Y as real variables    ''hex point 3 coordinates
55      define PT4.X and PT4.Y as real variables    ''hex point 4 coordinates
56      define PT5.X and PT5.Y as real variables    ''hex point 5 coordinates
57      define PT6.X and PT6.Y as real variables    ''hex point 6 coordinates
58      define SLOPE.A and SLOPE.F as real variables  ''slopes for hex sides A & F
59      define SLOPE.C and SLOPE.D as real variables  ''slopes for hex sides C & D
60      define Y.INT.A and Y.INT.F as real variables  ''y intercepts sides A & F
61      define Y.INT.C and Y.INT.D as real variables  ''y intercepts sides C & D
```

```
   every MINE has a MINE.X, a MINE.Y, a MINE.Z, a MINE.STATUS, a RADIUS,
     a MINE.TYPE, a MINE.NUM and belongs to the MINEFIELD

     define MINE.X, MINE.Y, and MINE.Z as real variables
     define RADIUS as a real variable
     define MINE.NUM as an integer variable
     define MINE.STATUS as an integer variable
     define MINE.TYPE as an integer variable

     define MINEFIELD as a set ranked by low MINE.Y
       The system owns the MINEFIELD

   every RED.OVERWATCH has a R.O.NUM, a R.O.TYPE, a R.O.RATE, a R.O.BETA,
     a R.O.STATUS, a R.O.X, and a R.O.Y

     define R.O.NUM as an integer variable      ''id
     define R.O.TYPE as an integer variable     ''type
     define R.O.RATE as a real variable         ''fire rate
     define R.O.BETA as a real variable         ''mean acquisition rate
     define R.O.STATUS as an integer variable   ''status
     define R.O.X and R.O.Y as real variables   ''location for display

   dynamic graphic entities include ELEMENT, MINE, OBSTACLE, RED.OVERWATCH

''====================== arrays ================================

                      '' movement maps of elements
   define MOVEMENT.PLAN as a 3-dimensional array
   define BYPASS.MAP as a 3-dimensional array

                      ''pk and fire allocation tables
   define MINE.PK as a 2-dimensional real array
   define R.O.PK as a 2-dimensional real array
   define FIRE.ALLOCATION as a 2-dimensional array

                      ''pointer arrays to access elements, obstacles,
                      ''red overwatch and mine objects.
   define E, OB, RO and M as 1-dimensional, pointer arrays

                      ''array of navigation checkpoint y values
   define CP as a 1-dimensional real array

                      ''these two arrays are used to keep track of
                      ''the type and ownership of encounter events
                      ''that are on the event calendar -- used when
                      ''it becomes necessary to clear the calendar.
   define EVENT.LIST as a 1-dimensional integer array
   define SCH.EVENTS as a 1-dimensional pointer array

   define TIME.OF.DEATH as a 1-dimensional double array
   define TIME.OF.COMPLETION as a 1-dimensional double array

''====================== global output variables ================

   define LINEAR.DENSITY as a real variable
   define LINEAR.DENSITY.TYPE.1 as a real variable
   define LINEAR.DENSITY.TYPE.2 as a real variable
   define LINEAR.DENSITY.TYPE.3 as a real variable
   define LINEAR.DENSITY.TYPE.4 as a real variable
   define LINEAR.DENSITY.TYPE.5 as a real variable

   define AREA.DENSITY as a real variable
```

```
125    define AREA.DENSITY.TYPE.1 as a real variable
126    define AREA.DENSITY.TYPE.2 as a real variable
127    define AREA.DENSITY.TYPE.3 as a real variable
128    define AREA.DENSITY.TYPE.4 as a real variable
129    define AREA.DENSITY.TYPE.5 as a real variable
130
131    define MINE.KILLS as a real variable
132    define MINE.KILL.1 as a real variable
133    define MINE.KILL.2 as a real variable
134    define MINE.KILL.3 as a real variable
135    define MINE.KILL.4 as a real variable
136    define MINE.KILL.5 as a real variable
137
138    define MINE.MOVED as a real variable
139
140    define R.O.SHOTS as a real variable
141    define R.O.KILL as a real variable
142
143    define LOST.TO.MINES.BYPASSING as a real variable
144    define LOST.TO.FIRES.BYPASSING as a real variable
145
146    define TOTAL.LOST as a real variable
147    define LOSS.RATE as a real variable
148
149  ''========== statistical outputs ==========
150
151    tally MEAN.MINE.KILLS as the mean, and VAR.MINE.KILLS
152      as the variance of MINE.KILLS
153    tally MEAN.MINE.KILL.1 as the mean, and VAR.MINE.KILL.1
154      as the variance of MINE.KILL.1
155    tally MEAN.MINE.KILL.2 as the mean, and VAR.MINE.KILL.2
156      as the variance of MINE.KILL.2
157    tally MEAN.MINE.KILL.3 as the mean, and VAR.MINE.KILL.3
158      as the variance of MINE.KILL.3
159    tally MEAN.MINE.KILL.4 as the mean, and VAR.MINE.KILL.4
160      as the variance of MINE.KILL.4
161    tally MEAN.MINE.KILL.5 as the mean, and VAR.MINE.KILL.5
162      as the variance of MINE.KILL.5
163
164    tally MEAN.MINE.MOVED as the mean and VAR.MINE.MOVED
165      as the variance of MINE.MOVED
166
167    tally MEAN.R.O.SHOTS as the mean and VAR.R.O.SHOTS as the
168      variance of R.O.SHOTS
169    tally MEAN.R.O.KILL as the mean and VAR.R.O.KILL as the
170      variance of R.O.KILL
171
172
173    tally MEAN.LOST.TO.MINES.BYPASSING as the mean and
174      VAR.LOST.TO.MINES.BYPASSING as the variance of
175      LOST.TO.MINES.BYPASSING
176
177    tally MEAN.LOST.TO.FIRES.BYPASSING as the mean and
178      VAR.LOST.TO.FIRES.BYPASSING as the variance of
179      LOST.TO.FIRES.BYPASSING
180
181    tally MEAN.LOSS.RATE as the mean and VAR.LOSS.RATE as the
182      variance of LOSS.RATE
183
184  ''==========global input variables==========
185
186    define NUMBER.OF.RUNS as an integer variable
```

```
187
188    define E.SPEED.1, E.SPEED.2, E.SPEED.3, E.SPEED.4, E.SPEED.5
189      as real variables
190    define E.WIDTH.1, E.WIDTH.2, E.WIDTH.3, E.WIDTH.4, E.WIDTH.5
191      as real variables
192    define E.LENGTH.1, E.LENGTH.2, E.LENGTH.3, E.LENGTH.4, E.LENGTH.5
193      as real variables
194    define E.TRACK.WIDTH.1, E.TRACK.WIDTH.2, E.TRACK.WIDTH.3,
195      E.TRACK.WIDTH.4, E.TRACK.WIDTH.5 as real variables
196    define NAV.ERR.1, NAV.ERR.2, NAV.ERR.3, NAV.ERR.4, NAV.ERR.5 as
197      real variables
198
199    define E.SPEED as a real variable
200
201    define MINEFIELD.DEPTH as a real variable
202    define NUM.ELEMENT, NUM.MINE and NUM.OBSTACLE as integer variables
203    define NUM.RED.OVERWATCH as an integer variable
204
205                                        ''amounts of different mines
206    define P.MINES, I.MINES, C.MINES, T.MINES,
207      and W.MINES as integer variables
208
209                                        ''sizes of different mines
210    define P.RADIUS, I.RADIUS, C.RADIUS, T.RADIUS,
211      and W.RADIUS as real variables
212
213                                        ''amounts of different vehicles
214    define TRACK1, TRACK2, TRACK3, F.PLOWS, and T.PLOWS as integer variables
215
216                                        ''characteristics of different plows
217    define PLOW.EFFECT.1 aa a real variable ''fw plow effectiveness
218    define PLOW.EFFECT.2 aa a real variable ''tw plow effectiveness
219    define EFF.WIDTH as a real variable     ''tw plow effective width
220    define PLOW.WIDTH.1 as a real variable  ''fw plow width
221    define PLOW.WIDTH.2 as a real variable  ''tw plow width
222    define PLOW.INTERVAL as a real variable ''interval between plows and
223                                            ''other taskforce elements
224
225    define R.O.RATE.1 as a real variable
226    define R.O.BETA.1 as a real variable
227
228    define NUM.CP as an integer variable
229    define INTERVAL as a real variable
230    define CP.INTERVAL aa a real variable
231
232    define GRAPH.ON as an integer variable
233
234                                        ''time variables
235    define minute to mean units
236    define minutes to mean units
237    define OLD.TIME aa a double variable
238
239    substitute theaa 5 lines for ..MOUSE.PAUSE ''to hold graphics on screen
240
241              call raadloc.r  given 0,0,0
242              Yielding DUMMY.X, DUMMY.Y, DUMMY.V
243              let DUMMY.X = DUMMY.X
244              let DUMMY.Y = DUMMY.Y
245              let DUMMY.V = DUMMY.V
246
247  end ''PREAMBLE
```

```
MAIN

define I as an integer variable

call DEFAULT.VALUES
call GET.DATA
call SET.DISPLAY

            ''turn graphics off for multiple runs
if NUMBER.OF.RUNS > 1
   GRAPH.ON = 0
endif

for I = 1 to NUMBER.OF.RUNS
  do
            ''this structure allows the user to display a particular
            ''run in a multiple run simulation. This might be done if
            ''the outputs appeared unusual or for random sampling of
            ''the model runs. Currently hardcoded for iteration 167,
            ''this can be turned into a global value and then set during
            ''the model initialization.
   if I = 167
     trace
     GRAPH.ON = 1
   endif

   call INITIALIZE

   LOSS.RATE = TOTAL.LOST / NUM.ELEMENT

            ''record desired iteration outputs
   call DATA.BIT giving I

loop

            ''record model run outputs
''call MINE.DUMP
''call DATA.DUMP
''call OUTPUT
call SUMMARY

stop

end ''MAIN
```

```
routine BYPASS given ID, OBS.ID, SIDE, X, Y

''BYPASS.MAP PLOTS A 4 POINT BYPASS ROUTE TO AVOID OBSTRUCTION IN MOVEMENT
''PATH. POINT 1 IS THE EXIT FROM THE MOVEMENT PATH. POINT 2 IS THE FIRST
''PIVOT, POINT 3 IS THE SECOND PIVOT, AND POINT 4 IS THE REENTRY POINT TO
''THE MOVEMENT PATH.

define ID as an integer variable
define OBS.ID as an integer variable
define SIDE as an integer variable
define X and Y as real variables
define COUNT as an integer variable        ''cp interval being examined
define SLOPE as a real variable
define SLOPE.2 as a real variable
define Y.INT as a real variable
define Y.INT.2 as a real variable
define INTERCEPT.X as a real variable
define INTERCEPT.Y as a real variable
define PIVOT2 as a real variable
define START.CP as a real variable

if SIDE = 0
  print 2 lines with SIDE thus
from BYPASS -- SIDE = *** --> error condition
changing SIDE to 1 to allow program continuation
  SIDE = 1
endif

        ''point 1
BYPASS.MAP(ID,1,1) = X    ''x value computed during distance.to.obs
BYPASS.MAP(ID,1,2) = Y    ''y value computed during distance.to.obs

        ''point 4 - usa point 4 of hex as temporary exit
        '' this valua will be ovarwrittan latter in this routine
BYPASS.MAP(ID,4,1) = PT4.X(OB(OBS.ID))
BYPASS.MAP(ID,4,2) = PT4.Y(OB(OBS.ID))

if (SIDE = 1)

  ''pass right

        ''point 1 - exit point
  BYPASS.MAP(ID,1,3) = SLOPE.A(OB(OBS.ID))
  BYPASS.MAP(ID,1,4) = Y.INT.A(OB(OBS.ID))

        ''point 2 - pivot 1
  BYPASS.MAP(ID,2,1) = PT2.X(OB(OBS.ID))    ''x of bypass pivot1
  BYPASS.MAP(ID,2,2) = PT2.Y(OB(OBS.ID))    ''y of bypass pivot1
  BYPASS.MAP(ID,2,3) = 1000000.0            ''approximate infinite slope
                                            ''y intercept
  BYPASS.MAP(ID,2,4) = BYPASS.MAP(ID,2,2) -
                       (BYPASS.MAP(ID,2,1) * BYPASS.MAP(ID,2,3))

        ''point 3 - pivot 2
  BYPASS.MAP(ID,3,1) = PT3.X(OB(OBS.ID))
  BYPASS.MAP(ID,3,2) = PT3.Y(OB(OBS.ID))
  BYPASS.MAP(ID,3,3) = (BYPASS.MAP(ID,3,2) - BYPASS.MAP(ID,4,2)) /
                       (BYPASS.MAP(ID,4,1) - BYPASS.MAP(ID,3,1))
  BYPASS.MAP(ID,3,4) = BYPASS.MAP(ID,3,2) -
                       (BYPASS.MAP(ID,3,1) * BYPASS.MAP(ID,3,3))
endif
```

```
63  if (SIDE = 2)
64    ''pass left
65
66          ''point 1
67    BYPASS.MAP(ID,1,3) = SLOPE.F(OB(OBS.ID))
68    BYPASS.MAP(ID,1,4) = Y.INT.F(OB(OBS.ID))
69
70          ''point 2
71    BYPASS.MAP(ID,2,1) = PT6.X(OB(OBS.ID))    ''x of bypass cp 2
72    BYPASS.MAP(ID,2,2) = PT6.Y(OB(OBS.ID))    ''y of bypass cp 2
73    BYPASS.MAP(ID,2,3) = 1000000.0            ''approximate infinite slope
74                                              ''y intercept
75    BYPASS.MAP(ID,2,4) = BYPASS.MAP(ID,2,2) -
76                        (BYPASS.MAP(ID,2,1) * BYPASS.MAP(ID,2,3))
77
78          ''point 3
79    BYPASS.MAP(ID,3,1) = PT5.X(OB(OBS.ID))
80    BYPASS.MAP(ID,3,2) = PT5.Y(OB(OBS.ID))
81    BYPASS.MAP(ID,3,3) = (BYPASS.MAP(ID,4,2) - BYPASS.MAP(ID,3,2)) /
82                         (BYPASS.MAP(ID,3,1) - BYPASS.MAP(ID,4,1))
83    BYPASS.MAP(ID,3,4) = BYPASS.MAP(ID,3,2) -
84                         (BYPASS.MAP(ID,3,1) * BYPASS.MAP(ID,3,3))
85  endif
86
87
88
89  ''CALCULATING the MOVEMENT.PLAN reentry Point.
90  '' SIDE = 1 indicates passing left
91  '' SIDE = 2 indicates passing right
92  ''  goal is to calculate the intersection point of the final bypass
93  ''  leg with the appropriate movement plan equation - the trick is to
94  ''  determine the appropriate movement plan equation - the intersection
95  ''  point must take place within the appropriate interval for that movment
96  ''  equation (determined by comparing the y value of the intercept with the
97  ''  upper boundary of the interval.
98
99
100 PIVOT2 = BYPASS.MAP(ID,3,2)  ''y coordinate of 2nd pivot
101 ''OB.END.Y = PT4.Y(OB(OBS.ID))
102 START.CP = 0
103                               ''sanity check
104 if PIVOT2 > CP(NUM.CP)
105   print 1 line thus
pivot2 extends beyond boundary of problem
107   ''should terminate at intersection of 2nd bypass leg and boundary
108 endif
109
110                               ''find interval which contains pivot 2
111 for I = 1 to NUM.CP
112   do
113   if CP(I) < PIVOT2
114      START.CP = I
115   endif
116 loop
117
118
119                               ''not currently in final interval
120 if ((START.CP + 1) ne NUM.CP)
121
122   for COUNT = START.CP to NUM.CP-1
123     do
124
```

```
125      SLOPE = MOVEMENT.PLAN(ID,COUNT,3) ''m1 for movement path segment
126      Y.INT = MOVEMENT.PLAN(ID,COUNT,4) ''b1 intercept for same segment
127
128      if (SIDE = 1)
129        SLOPE.2 = SLOPE.D(OB(OBS.ID))   ''m2 for bypass segment left
130        Y.INT.2 = Y.INT.D(OB(OBS.ID))
131        INTERCEPT.X = (Y.INT.2 - Y.INT) / (SLOPE - SLOPE.2)
132        INTERCEPT.Y = SLOPE * INTERCEPT.X + Y.INT
133
134
135      endif
136
137      if (SIDE = 2)
138        SLOPE.2 = SLOPE.C(OB(OBS.ID))   ''m2 for bypass segment right
139        Y.INT.2 = Y.INT.C(OB(OBS.ID))   ''b2 for bypass segment right
140        INTERCEPT.X = (Y.INT.2 - Y.INT) / (SLOPE - SLOPE.2)
141        INTERCEPT.Y = SLOPE * INTERCEPT.X + Y.INT
142
143      endif
144
145                                        ''condition for forward movement
146      if (INTERCEPT.Y > BYPASS.MAP(ID,3,2))
147                                        ''intersection occurs within the
148                                        ''boundary of the interval being
149                                        ''examined
150        if(INTERCEPT.Y < MOVEMENT.PLAN(ID, COUNT+1, 2))
151          BYPASS.MAP(ID,4,1) = INTERCEPT.X
152          BYPASS.MAP(ID,4,2) = INTERCEPT.Y
153          leave
154        endif
155      endif
156
157     loop
158
159    else ''reenter at exit boundary
160
161     BYPASS.MAP(ID,4,2) = CP(NUM.CP)
162     BYPASS.MAP(ID,4,1) = (BYPASS.MAP(ID,4,2) - BYPASS.MAP(ID,3,4)) /
163                          BYPASS.MAP(ID,3,3)
164  endif
165
166  'DONE'
167
168  return
169
170  end ''BYPASS   edited 31 AUG 91
```

```
routine CALENDAR.UPDATE

''THIS ROUTINE IS CALLED ANYTIME AN ELEMENT IS CONVERTED INTO AN OBSTACLE. ITS
''PURPOSE IS TO PREVENT A 'TIME-WARP', WHERE AN ALREADY SCHEDULED EVENT
''IS CARRIED OUT EVEN THOUGH THE SITUATION HAS CHANGED AND THE DATA UPON
''WHICH THAT EVENT WAS ORIGINALLY SCHEDULED HAS CHANGED. THE ROUTINE CANCELS
''ALL ENCOUNTER EVENTS, AND THEN RESCHEDULES EACH ACTIVE ELEMENT BASED UPON
''THE DATA EXISTANT WHEN THIS ROUTINE WAS CALLED

define I as an integer variable

for I = 1 to NUM.ELEMENT
  do

  select case EVENT.LIST(I)

      case 1

          cancel the NEW.CP called SCH.EVENTS(I)
          EVENT.LIST(I) = 0

      case 2

          cancel the MINE.ENCOUNTER called SCH.EVENTS(I)
          EVENT.LIST(I) = 0

      case 3

          cancel the OBSTACLE.ENCOUNTER called SCH.EVENTS(I)
          EVENT.LIST(I) = 0

      case 0

  endselect

loop

for I = 1 to NUM.ELEMENT
  do
  if ((STATUS(E(I)) ne 0) and (SPEED(E(I)) ne 0))

     call NEXT.ENCOUNTER giving I

  endif
loop

return

end ''CALENDAR UPDATE edited 28 July 91
```

```
    1  routine DATA.BIT given I
    2
    3  ''THIS ROUTINE USED TO RECORD INFORMATION DESIRED FROM EACH ITERATION
    4
    5  define I as an integer variable
    6
    7  print 1 line with I, MINE.KILLS, R.O.KILL, TOTAL.LOST, LOST.TO.MINES.BYPASSING,
    8    TIME.OF.DEATH(1), TIME.OF.DEATH(2), TIME.OF.DEATH(3), TIME.OF.DEATH(4),
    9    TIME.OF.DEATH(5), TIME.OF.DEATH(6), TIME.OF.COMPLETION(1),
   10    max.f(TIME.OF.COMPLETION(1), TIME.OF.COMPLETION(2), TIME.OF.COMPLETION(3),
   11    TIME.OF.COMPLETION(4), TIME.OF.COMPLETION(5), TIME.OF.COMPLETION(6)) thus
*** ** ** ** ** ***.** ***.** ***.** ***.** ***.** ***.** ***.** ***.** ***.**
   13
   14  return
   15  end ''DATA.BIT
```

```
    1  routine DATA.DUMP
    2
    3  ''THIS ROUTINE CAUSES THE GENERATED MOVEMENT PLAN ARRAYS TO BE RECORDED
    4  ''AND SUMMARIZES THE VEHICLE STATUS AND OBSTACLE STATUS AND THE END OF
    5  ''AN ITERATION -- SHOULD NOT BE USED FOR MULTIPLE ITERATIONS
    6
    7  define I as an integer variable
    8  define CONDITION as a text variable
    9
   10  print 3 lines thus

        DUMP OF VEHICLE RELATED DATA

   14
   15  print 5 lines with time.v thus
               ELEMENT STATUS AT
                TIME = ****.***

  ID     X          Y      SPEED  STATUS  TYPE
  --  -------    ------- ------- ------  ----
   21
   22  for I = 1 to NUM.ELEMENT
   23    do
   24    print 1 line with ELEMENT.NUM(E(I)), ELEMENT.X(E(I)), ELEMENT.Y(E(I)),
   25      SPEED(E(I)), STATUS(E(I)), and ELEMENT.TYPE(E(I)) thus
  **  ****.**   ****.** ***.***    **    **
   27  loop
   28
   29  print 4 lines thus

               OBSTACLE STATUS
    NUM        X         Y      RADIUS
    ---     -------   -------   ------
   34
   35  for j = 1 to NUM.OBSTACLE
   36  do
   37     print 1 line with J, OBSTACLE.X(OB(J)), OBSTACLE.Y(OB(J)),
   38          OBS.RADIUS(OB(J)) thus
    **      ***.***   ***.***   ***.**
   40  loop
   41
   42  print 3 lines thus

BYPASS.MAP
  ELE  E-X    E-Y   ST   P1-X   P1-Y  ST  P2-X   P2-Y  ST Ent-X  Ent.Y   ST
   46  for J = 1 to NUM.ELEMENT
   47    do
   48    print 1 line with J,BYPASS.MAP(J,1,1), BYPASS.MAP(J,1,2), BYPASS.MAP(J,1,5),
   49       BYPASS.MAP(J,2,1), BYPASS.MAP(J,2,2), BYPASS.MAP(J,2,5),
   50       BYPASS.MAP(J,3,1), BYPASS.MAP(J,3,2), BYPASS.MAP(J,3,5),
   51       BYPASS.MAP(J,4,1), BYPASS.MAP(J,4,2), BYPASS.MAP(J,4,5) thus
  **  **.**  **.**  **  **.**  **.**  **  **.**  **.** **  **.**  **.**  **
   53    loop
   54
   55  for J = 1 to NUM.ELEMENT
   56    do
   57    print 1 line with J, ELEMENT.TYPE(E(J)) thus
  ELEMENT # ** has type **
   59
   60    print 2 lines thus

      X-CORR  Y-CORR  SLOPE  INTERCEPT
```

```
   63
   64    for I = 1 to NUM.CP
   65      do
   66        print 1 line with MOVEMENT.PLAN(J,I,1), MOVEMENT.PLAN(J,I,2),
   67           MOVEMENT.PLAN(J,I,3), MOVEMENT.PLAN(J,I,4) thus
      ***.***, ***.**, ***.***, ***.***
   69        start new output line
   70    loop
   71  loop
   72
   73   print 4 lines thus
        ELEMENT STATUS
        ==============
 NUM  STATUS     X        Y
 ---  ------  -------  -------
   78
   79   for J = 1 to NUM.ELEMENT
   80     do
   81       if STATUS(E(J)) = 0
   82           CONDITION = "DEAD "
   83       else
   84           CONDITION = "ALIVE"
   85       endif
   86     print 1 line with J, CONDITION, ELEMENT.X(E(J)), ELEMENT.Y(E(J)) thus
**  *****   ***.***   ***.***
   88   loop
   89
   90   print 4 lines thus
     OBSTACLE LOCATIONS
     ==================
 NUM       X         Y
 ---   --------- ----------
   95
   96   for J = 1 to NUM.OBSTACLE
   97     do
   98       if OBSTACLE.Y(OB(J)) > 0
   99         print 1 line with J, OBSTACLE.X(OB(J)), OBSTACLE.Y(OB(J)) thus
**     ***.****  ***.****
  101       endif
  102   loop
  103
  104  return
  105  end ''DATA.DUMP
```

```
routine DEFAULT.VALUES

''THIS ROUTINE SETS DEFAULT VALUES FOR ALL INPUT VARIABLES.

reserve MINE.PK(*,*) as 5 by 5

''default values for TASKFORCE menu
''--------------------------------
F.PLOWS = 1
T.PLOWS = 0
TRACK1  = 5
TRACK2  = 0
TRACK3  = 0


''default values for Type 1 vehicle menu (fw plow)
''-----------------------------------------------
E.WIDTH.1       = 3.48
E.LENGTH.1      = 9.03
E.TRACK.WIDTH.1 = .635
NAV.ERR.1       = 1.0


''Default values for full width plow specification input form
''==========================================
PLOW.EFFECT.1   = 1.0
PLOW.WIDTH.1    = 4.5
E.SPEED.1       = 6.0



''default values for Type 2 vehicle menu
''--------------------------------------
E.WIDTH.2       = 3.48
E.LENGTH.2      = 9.03
E.TRACK.WIDTH.2 = .635
NAV.ERR.2       = 1.0

''Default values for track width plow specification input form
''============================================================
PLOW.EFFECT.2   = 1.0
PLOW.WIDTH.2    = 1.0
E.SPEED.2       = 6.0
EFF.WIDTH       = 4.0


''default values for Type 3 vehicle menu
''--------------------------------------
E.WIDTH.3       = 3.48
E.LENGTH.3      = 9.03
E.TRACK.WIDTH.3 = .635
NAV.ERR.3       = .5


''default values for Type 4 vehicle menu
''--------------------------------------
E.WIDTH.4       = 3.48
E.LENGTH.4      = 9.03
E.TRACK.WIDTH.4 = .635
NAV.ERR.4       = 1.0


```

```
''default values for Type 5 vehicle menu
''-------------------------------------
E.WIDTH.5       = 3.48
E.LENGTH.5      = 9.03
E.TRACK.WIDTH.5 = .635
NAV.ERR.5       = 1.0


''default values for Mine Data menu
''---------------------------------

P.MINES = 50
I.MINES = 30
C.MINES = 8
T.MINES = 0
W.MINES = 0



''Default values for mine specification input form
''===============================================

P.RADIUS  = .1
I.RADIUS  = .1
C.RADIUS  = .1
T.RADIUS  = 0
W.RADIUS  = 0

MINE.PK(1,1)  = .05
MINE.PK(1,2)  = .1
MINE.PK(1,3)  = .9
MINE.PK(1,4)  = 0
MINE.PK(1,5)  = 0

MINE.PK(2,1)  = .05
MINE.PK(2,2)  = .5
MINE.PK(2,3)  = .9
MINE.PK(2,4)  = 0
MINE.PK(2,5)  = 0

MINE.PK(3,1)  = .5
MINE.PK(3,2)  = .7
MINE.PK(3,3)  = .9
MINE.PK(3,4)  = 0
MINE.PK(3,5)  = 0

MINE.PK(4,1)  = 0
MINE.PK(4,2)  = 0
MINE.PK(4,3)  = 0
MINE.PK(4,4)  = 0
MINE.PK(4,5)  = 0

MINE.PK(5,1)  = 0
MINE.PK(5,2)  = 0
MINE.PK(5,3)  = 0
MINE.PK(5,4)  = 0
MINE.PK(5,5)  = 0



''Default values for red overwatch form
''=====================================
```

```
  125
  126  NUM.RED.OVERWATCH = 1
  127  R.O.RATE.1 = .5
  128  R.O.BETA.1 = .7
  129
  130  return
  131
  132  end ''DEFAULT.VALUES
```

```
 1  routine DELTA.DISTANCE given ID,TRAVEL.TIME yielding DISTANCE
 2
 3  ''DETERMINES HOW FAR THE ID TRAVELED IN THE GIVEN TIME
 4
 5  define ID as an integer variable
 6  define TRAVEL.TIME as a double variable
 7  define DISTANCE as a real variable
 8  define VELOCITY as a real variable
 9
10  VELOCITY = SPEED(E(ID))
11  DISTANCE = TRAVEL.TIME * VELOCITY
12
13  return
14
15  end ''DELTA.DISTANCE
```

```
    1  routine DELTA.TIME given ID and DISTANCE yielding DURATION
    2
    3  ''THIS ROUTINE DETERMINES HOW LONG IT WILL TAKE AN ELEMENT TO COVER
    4  ''A GIVEN DISTANCE. THE ROUTINE ACCESSES THE ELEMENT RECORD TO
    5  ''DETERMINE ELEMENT SPEED AND RETURNS TO THE CALLER THE TIME IT TAKES
    6  ''THAT ELEMENT TO COVER THAT DISTANCE
    7
    8  define ID as an integer variable
    9  define DISTANCE as a real variable
   10  define DURATION as a double variable
   11
   12  DURATION = DISTANCE / SPEED(E(ID))
   13
   14  return
   15
   16  end ''DELTA.TIME
```

```
 1  event DIRECT.FIRE given ID
 2
 3  ''THIS EVENT RESOLVES A DIRECT FIRE ENGAGEMENT BETWEEN A RED OVERWATCH
 4  ''ELEMENT AND A LIVING ELEMENT IN THE MINEFIELD. HOOKS EXIST FOR ALSO
 5  ''ALLOWING DIRECT FIRE ENGAGEMENTS AGAINST ALREADY DISABLED VEHICLES,
 6  ''BUT THIS OPTION IS NOT YET IMPLEMENTED.
 7
 8
 9  define ID as an integer variable            ''firer id
10  define I as an integer variable             ''counter
11  define COUNT as an integer variable         ''number of targets
12  define DEAD.COUNT as an integer variable    ''number of dead targets
13  define TGT.LIST as a 1-dimensional integer array
14  define DEAD.TGT.LIST as a 1-dimensional integer array
15  define P.TGT as a 1-dimensional real array  ''prob of shooting this tgt
16  define TGT as an integer variable           ''target id
17  define P1, P2, P3 as a real variables
18  define PK as a real variable                ''pk of killing if shot
19  define ROLL as a real variable              ''dice roll
20  define CUM.PROB as a real variable
21  define FIRST as an integer variable         ''pointer to first active
22                                              ''  element in minefield
23  define X, Y as real variables               ''firer display location
24  define X1, Y1 as real variables
25  define Y.MIN, Y.MAX as real variables
26  define FLAG as an integer variable          ''any moving vehicles ?
27
28
29  reserve TGT.LIST(*) as NUM.ELEMENT
30  reserve DEAD.TGT.LIST(*) as NUM.ELEMENT
31  reserve P.TGT(*) as NUM.ELEMENT
32
33
34  COUNT = 0
35  DEAD.COUNT = 0
36  PK = .2
37  X = R.O.X(RO(ID))
38  Y = R.O.Y(RO(ID))
39
40  R.O.SHOTS = R.O.SHOTS + 1.0
41
42                          ''determine how many targets exist -
43                          ''target is defined as active element,
44                          ''which has traveled at least 15 meters from
45                          ''start point (arbitrary value) and is futher
46                          ''than 15 meters from finish point. Disabled
47                          ''vehicles may also be included if desired.
48  FLAG = 0
49  Y.MIN = 15.0
50  Y.MAX = (NUM.CP * CP.INTERVAL) - 15.0
51
52  for I = 1 to NUM.ELEMENT
53    do
54    if (SPEED(E(I)) > 0)
55      if ELEMENT.Y(E(I)) > Y.MIN
56        if ELEMENT.Y(E(I)) < Y.MAX
57          TGT.LIST(I) = 1
58          FLAG = 1
59          COUNT = COUNT + 1
60        endif
61      endif
62    endif
```

```
   63
   64    if OBSTACLE.Y(OB(I)) > 0
   65      DEAD.TGT.LIST(I) = 1
   66      DEAD.COUNT = DEAD.COUNT + 1
   67    endif
   68  loop
   69                        ''no targets
   70  if COUNT = 0
   71    go to 'BOTTOM'
   72  endif
   73
   74  select case COUNT
   75    case 0
   76      P1 = 0
   77    case 1
   78      P1 = 1.0
   79    case 2
   80      P1 = .8
   81    case 3
   82      P1 = .7
   83    case 4
   84      P1 = .6
   85    case 5
   86      P1 = .5
   87    default
   88      P1 = .4
   89      print 1 line thus
    p1 defaulted to a value of .4
   91  endselect
   92
   93  ROLL = uniform.f(0.0, 1.0, 4)
   94
   95  P2 = 1 - P1
   96
   97  if COUNT > 1
   98    P3 = P2/(COUNT - 1)
   99  endif
  100
  101  CUM.PROB = 0
  102  FIRST = 0
  103
  104  for I = 1 to NUM.ELEMENT
  105    do
  106    if TGT.LIST(I) > 0
  107      if FIRST = 0
  108        FIRST = 1
  109        P.TGT(I) = P1
  110        CUM.PROB = CUM.PROB + P1
  111       else
  112        P.TGT(I) = CUM.PROB + P3
  113        CUM.PROB = CUM.PROB + P3
  114      endif
  115    endif
  116  loop
  117
  118  for I = 1 to NUM.ELEMENT
  119    do
  120    if ROLL < P.TGT(I)
  121      TGT = I
  122      leave
  123    endif
  124  loop
```

```
  125                              ''determine location of target
  126  X1 = ELEMENT.X(E(TGT))
  127  Y1 = ELEMENT.Y(E(TGT))
  128
  129  if GRAPH.ON = 1
  130    show RO(ID) with "rotankf.icn"
  131
  132    if STATUS(E(TGT)) = 0
  133      show E(TGT) with "tanki.icn"
  134    endif
  135
  136    if STATUS(E(TGT)) = 1
  137
  138      if ELEMENT.TYPE(E(TGT)) = 1
  139        show E(TGT) with "fwplowi.icn"
  140      endif
  141
  142      if ELEMENT.TYPE(E(TGT)) = 3
  143        show E(TGT) with "tank1i.icn"
  144      endif
  145
  146    endif
  147
  148    let location.a(RO(ID)) = location.f(X,Y)
  149    let location.a(E(TGT)) = location.f(X1,Y1)
  150  endif
  151              ''pk is the probability of kill given a shot
  152  ROLL = uniform.f(0,1,7)
  153
  154  PK = .2
  155
  156  if ROLL < PK
  157    if STATUS(E(TGT)) ne 0
  158
  159      TOTAL.LOST = TOTAL.LOST + 1.0
  160      R.O.KILL = R.O.KILL + 1.0
  161
  162      if STATUS(E(TGT)) = 2
  163        LOST.TO.FIRES.BYPASSING = LOST.TO.FIRES.BYPASSING + 1
  164      endif
  165
  166      STATUS(E(TGT)) = 0
  167      SPEED(E(TGT)) = 0
  168      if TIME.OF.DEATH(TGT) > 0
  169        print 2 lines with TGT, TIME.OF.DEATH(TGT) thus
     #### ERROR IN DIRECT FIRE -- KILLED DEAD TARGET
           TGT *** was recorded as killed at time ****.****
  172      endif
  173
  174      TIME.OF.DEATH(TGT) = time.v
  175
  176      if GRAPH.ON = 1
  177        if ELEMENT.TYPE(E(I)) = 1
  178          show E(TGT) with "dfwplow.icn"
  179         else
  180          show E(TGT) with "tank.icn"
  181        endif
  182        let location.a(E(TGT)) = location.f(X1,Y1)
  183      endif
  184
  185      OBSTACLE.X(OB(TGT)) = X1
  186      OBSTACLE.Y(OB(TGT)) = Y1
```

```
  187      OBS.RADIUS(OB(TGT)) = ELEM.RADIUS(E(TGT))
  188
  189      call OBSTACLE.CONSOLIDATION
  190      call CALENDAR.UPDATE
  191
  192    endif
  193  endif
  194
  195              ''if there are still moving Blue vehicles,
  196              ''schedule another shot for this firer
  197  if FLAG = 1
  198   call R.DIRECT.OVERWATCH giving ID
  199  endif
  200
  201  'BOTTOM'
  202
  203  return
  204
  205  end ''DIRECT.FIRE
```

```
 1  routine DISTANCE.TO.CP given ID yielding DISTANCE, DEL.X, DEL.Y
 2
 3  ''THIS ROUTINE DETERMINES THE DISTANCE BETWEEN THE IDENTIFIED ELEMENT
 4  ''AND THE NEXT CHECKPOINT ON THAT ELEMENTS MOVEMENT PLAN.
 5
 6  define ID as an integer variable        ''element index
 7  define CURR.CP as an integer variable   ''checkpoint index
 8  define DISTANCE as real variable        ''distance between element & cp
 9  define DEL.X as a real variable         ''difference in x
10  define DEL.Y as a real variable         ''difference in y
11
12                                  ''determine element's current location
13                                  ''on its respective movement plan, then
14  call FIND.CURRENT.CP giving ID yielding CURR.CP
15
16  if STATUS(E(ID)) = 1
17                                  ''calculate the x distance between the
18                                  ''next checkpoint and the element, then
19    DEL.X = MOVEMENT.PLAN(ID,CURR.CP + 1,1) - ELEMENT.X(E(ID))
20
21                                  ''calculate the y distance between the
22                                  ''next checkpoint and the element, then
23    DEL.Y = MOVEMENT.PLAN(ID,CURR.CP + 1,2) - ELEMENT.Y(E(ID))
24
25                                  ''use the pythagorium theorem to determine
26                                  ''the straight line distance between the
27                                  ''element and the next checkpoint and
28
29  endif
30
31  if STATUS(E(ID)) = 2
32
33    DEL.X = BYPASS.MAP(ID,CURR.CP + 1,1) - ELEMENT.X(E(ID))
34    DEL.Y = BYPASS.MAP(ID,CURR.CP + 1,2) - ELEMENT.Y(E(ID))
35
36  endif
37                                  ''compute the distance between the points
38    DISTANCE = sqrt.f(DEL.X**2 + DEL.Y**2)
39
40
41  return
42
43  end ''DISTANCE.TO.CP
```

```
routine DISTANCE.TO.MINE given ID yielding MINE.ID, RANGE.TO.MINE

''THIS ROUTINE DETERMINES THE CLOSEST MINE THAT AN ELEMENT WILL ACTUALLY
''HIT. INPUTS ARE THE ELEMENT ID. THE ROUTINE RETURNS THE DISTANCE FROM THE
''ELEMENT TO THE NEXT MINE THAT WILL BE HIT.

define ID as an integer variable
define SLOPE as a real variable
define INTERCEPT as a real variable
define MINE.SLOPE as a real variable
define Y.BOTTOM as a real variable
define Y.TOP as a real variable
define X.LEFT as a real variable
define X.RIGHT as a real variable
define MINE.ID as an integer variable        '' id of closest mine
define POSSIBLE.MINE as an integer variable '' id of candidate mine
define DISTANCE.TO.MINE.ENC as a real variable
define RANGE.TO.MINE as a real variable
define POSSIBLE.RANGE as a real variable
define CURR.CP as an integer variables
define K as an integer variables
define X1,X2,X3,Y1,Y2,Y3 as real variables
define MISS.DIS as a real variable           ''distance of mine center from
                                             ''    element path
define TRACK as a real variable              ''width of single element track
define E.WIDTH as a real variable            ''effective element width

                          ''if no mines ara found, raturn distance of
                          ''1000 matars and a mine id of 0
RANGE.TO.MINE = 1000.0
POSSIBLE.RANGE = 1000.0
MINE.ID = 0
TRACK = TRACK.WIDTH(E(ID))

call FIND.CURRENT.CP giving ID yielding CURR.CP
if (CURR.CP ne NUM.CP)
                          ''record path equation data

  SLOPE = MOVEMENT.PLAN(ID,CURR.CP,3)
  INTERCEPT = MOVEMENT.PLAN(ID,CURR.CP,4)

                          ''if element is currently bypassing, use the
                          ''bypass map for path equation data.
  if STATUS(E(ID)) = 2
    SLOPE = BYPASS.MAP(ID,CURR.CP,3)
    INTERCEPT = BYPASS.MAP(ID,CURR.CP,4)
  endif

                          ''set filters
                          ''filters are used to eliminate
                          ''from inspection those mines that
                          ''are too far away to be possible
                          ''encounters. This reduces the number
                          ''of calculations required.
  Y.BOTTOM = ELEMENT.Y(E(ID))
  Y.TOP = ELEMENT.Y(E(ID)) + (1.2* CP.INTERVAL)

  if (MOVEMENT.PLAN(ID,CURR.CP,1) < MOVEMENT.PLAN(ID,CURR.CP+1,1))
    X.LEFT = MOVEMENT.PLAN(ID,CURR.CP,1) - 12.0
    X.RIGHT = MOVEMENT.PLAN(ID,CURR.CP+1,1) + 12.0
   else
    X.LEFT = MOVEMENT.PLAN(ID,CURR.CP+1,1) - 12.0
```

```
    X.RIGHT = MOVEMENT.PLAN(ID,CURR.CP,1) + 12.0
  endif

                          ''if element is in bypass mode
                          ''adjust filters accordingly
  if (STATUS(E(ID)) = 2),
    if(BYPASS.MAP(ID,CURR.CP,1) <= BYPASS.MAP(ID,CURR.CP+1,1)),
      X.LEFT = BYPASS.MAP(ID,CURR.CP,1) - 12.0
      X.RIGHT = BYPASS.MAP(ID,CURR.CP+1,1) + 12.0
     else
      X.LEFT = BYPASS.MAP(ID,CURR.CP+1,1) - 12.0
      X.RIGHT = BYPASS.MAP(ID,CURR.CP,1) + 12.0
    endif
  endif
                          ''use filters to reduce the number of mines
                          ''examined by looking at each mine
  for K = 1 to NUM.MINE
    do
    if MINE.STATUS(M(K)) > 0
      if MINE.Y(M(K)) > Y.BOTTOM
        if MINE.Y(M(K)) < Y.TOP
          if MINE.X(M(K)) > X.LEFT
            if MINE.X(M(K)) < X.RIGHT

                                ''calculate slope of line perpindicular
                                ''to path equation using M1*M2=-1
              MINE.SLOPE = -1.0 / SLOPE
                                ''use point-slope formula,
                                ''y-y1=m(x-x1) to develop equation for
                                ''mine-line, combine with equation for
                                ''movement path line to solve for
                                ''intersection point
              X1 = (MINE.Y(M(K)) - INTERCEPT - (MINE.SLOPE * MINE.X(M(K)))) /
                        (SLOPE - MINE.SLOPE)
              Y1 = X1 * SLOPE + INTERCEPT

                                ''compute mine distance from movement path
              X2 = X1 - MINE.X(M(K))
              Y2 = Y1 - MINE.Y(M(K))
              MISS.DIS = SQRT.F(X2**2 + Y2**2)

                                ''compute mine encounter distance
                                ''from moving element
              X3 = X1 - ELEMENT.X(E(ID))
              Y3 = Y1 - ELEMENT.Y(E(ID))
              DISTANCE.TO.MINE.ENC = SQRT.F(X3**2 + Y3**2)


              E.WIDTH = .5 * max.f(WIDTH(E(ID)),PLOW.WIDTH(E(ID)))
              MISS.DIS = MISS.DIS - RADIUS(M(K))

              if (ELEMENT.TYPE(E(ID)) = 1) and (MISS.DIS < E.WIDTH)
                POSSIBLE.MINE = K
                POSSIBLE.RANGE = DISTANCE.TO.MINE.ENC
              endif

              if ELEMENT.TYPE(E(ID)) ne 1

                select case MINE.TYPE(M(K))

                case 1        '' influence mine
                  if (MISS.DIS < E.WIDTH)
```

```
125                     POSSIBLE.MINE = K
126                     POSSIBLE.RANGE = DISTANCE.TO.MINE.ENC
127                   endif
128
129                 case 2       '' pressure mine
130                  if ((MISS.DIS <  E.WIDTH) and (MISS.DIS > (E.WIDTH - TRACK)))
131                     POSSIBLE.MINE = K
132                   POSSIBLE.RANGE = DISTANCE.TO.MINE.ENC
133                 endif
134
135               case 3       '' contact mine
136                 if (MISS.DIS < E.WIDTH)
137                   POSSIBLE.MINE = K
138                   POSSIBLE.RANGE = DISTANCE.TO.MINE.ENC
139                 endif
140
141             endselect
142           endif
143
144           if POSSIBLE.RANGE < RANGE.TO.MINE,
145              MINE.ID = POSSIBLE.MINE
146              RANGE.TO.MINE = POSSIBLE.RANGE
147           endif
148         endif
149       endif
150     endif
151   endif
152 endif
153
154 loop
155
156 endif
157
158 return
159
160 end ''DISTANCE.TO.MINE edited 31 July 91
```

```
 1  function DISTANCE.TO.OBS given ID yielding DISTANCE
 2
 3  ''DETERMINES THE DISTANCE TO THE NEXT OBSTACLE FOR THE GIVEN ELEMENT
 4
 5  define ID as an integer variable
 6  define DISTANCE as a real variable
 7  define OBS.ID as an integer variable
 8  define E.X and E.Y as real variables
 9  define CURR.CP as an integer variable
10  define C.Y as a real variable
11  define SLOPE as a real variable
12  define Y.INT as a real variable
13  define K as an integer variable
14  define RANGE.TO.OBS as a real variable
15  define CANDIDATE.OBS as an integer variable
16  define PT1Y as a real variable
17  define PT2Y as a real variable
18  define PT1X as a real variable
19  define PT6X as a real variable
20  define PT2X as a real variable
21  define SLP.A as a real variable
22  define SLP.F as a real variable
23  define Y.I.A as a real variable
24  define Y.I.F as a real variable
25  define INTERCEPT.X.1 as a real variable
26  define INTERCEPT.X.2 as a real variable
27  define X, Y as real variables
28  define SIDE as an integer variable          ''flag passed to bypass - indicates
29                                              ''which side to pass on
30  DISTANCE = 10000.0
31  RANGE.TO.OBS = 10000.0
32  OBS.ID = 1
33
34  if STATUS(E(ID)) ne 2
35                                  ''determine what portion of map is being used
36    call FIND.CURRENT.CP giving ID yielding CURR.CP
37
38    E.X   = ELEMENT.X(E(ID))
39    E.Y   = ELEMENT.Y(E(ID))
40    C.Y   = MOVEMENT.PLAN(ID,CURR.CP+1,2)
41    SLOPE = MOVEMENT.PLAN(ID,CURR.CP,3)  ''m1
42    Y.INT = MOVEMENT.PLAN(ID,CURR.CP,4)  ''b1
43
44    for K = 1 to NUM.OBSTACLE
45      do
46                                  ''if obstruction exists, build hex for it
47      if OBS.RADIUS(OB(K)) > 0
48        call MAKE.HEX giving ID, K
49        PT1Y = PT1.Y(OB(K))
50        PT2Y = PT2.Y(OB(K))
51
52                                  ''determine is obstacle can be encountered
53                                  ''in current interval.
54        if (PT2Y > E.Y)
55          if (PT1Y < C.Y)
56                                  ''determine intercept point, check for encounter
57            PT6X = PT6.X(OB(K))
58            PT1X = PT1.X(OB(K))
59            PT2X = PT2.X(OB(K))
60
61            SLP.A = SLOPE.A(OB(K)) ''m2
62            SLP.F = SLOPE.F(OB(K)) ''m2
```

```
63            Y.I.A = Y.INT.A(OB(K)) ''b2
64            Y.I.F = Y.INT.F(OB(K)) ''b2
65
66            INTERCEPT.X.1 = (Y.I.A - Y.INT) / (SLOPE - SLP.A)
67            INTERCEPT.X.2 = (Y.I.F - Y.INT) / (SLOPE - SLP.F)
68
69            if (PT1X < INTERCEPT.X.1) and (INTERCEPT.X.1 < PT2X)
70              X = INTERCEPT.X.1
71              Y = (SLP.A * X) + Y.I.A
72              CANDIDATE.OBS = K
73              RANGE.TO.OBS = sqrt.f((X - E.X)**2 + (Y - E.Y)**2)
74              SIDE = 1
75            endif
76
77            if (PT6X < INTERCEPT.X.2) and (INTERCEPT.X.2 < PT1X)
78              X = INTERCEPT.X.2
79              Y = (SLP.F * X) + Y.I.F
80              CANDIDATE.OBS = K
81              RANGE.TO.OBS = sqrt.f((X - E.X)**2 + (Y - E.Y)**2)
82              SIDE = 2
83            endif
84          endif
85        endif
86      endif
87
88      if RANGE.TO.OBS < DISTANCE
89        DISTANCE = RANGE.TO.OBS
90        OBS.ID = CANDIDATE.OBS
91        call BYPASS giving ID, OBS.ID, SIDE, X, Y
92      endif
93
94    loop
95
96  endif
97
98  return
99
100 end ''DISTANCE.TO.OBS  edited 25 AUG 91
```

```
routine FIND.CURRENT.CP given ID yielding CURR.CP

''THIS ROUTINE DETERMINES WHICH CHECKPOINT ON THE MOVEMENT.PLAN
''IS CURRENT FOR A GIVEN ELEMENT

define ID as an integer variables
define CURR.CP as an integer variable
define J as an integer variable

if ((STATUS(E(ID)) = 1) or (STATUS(E(ID)) = 0)),
  for J = 1 to NUM.CP do          ''check each checkpoint in order,
                                  ''looking for the current one for
                                  ''element ID.
    if MOVEMENT.PLAN(ID,J,5) > 0,
                                  ''when found, record the index
      CURR.CP = J
                                  ''and exit the loop
      leave
    endif
 loop
endif

if STATUS(E(ID)) = 2
  for J = 1 to 4 do
    if BYPASS.MAP(ID,J,5) > 0,     ''look for active status flag
                                   ''{bypass.map(*,*,5) > 0}

      CURR.CP = J                 ''when found, record that point and
      leave                       ''exit loop
    endif
  loop
endif

return                            ''return current cp index

end ''FIND.CURRENT.CP
```

```
routine FULL.PLOW given ID, MINE.ID

''THIS ROUTINE MOVES MINES ENCOUNTERED BY FULL PLOWS

define ID as an integer variable
define MINE.ID as an integer variable
define X.ID as real variables
define X.M, Y.M as real variables
define X, Y as real variables
define NEW.X.M, NEW.Y.M as real variables
define PLOW.WID as a real variable
define DISPLACEMENT as a real variable

X.ID = ELEMENT.X(E(ID))
X.M = MINE.X(M(MINE.ID))
Y.M = MINE.Y(M(MINE.ID))
PLOW.WID = PLOW.WIDTH(E(ID))/2.0

if MINE.STATUS(M(MINE.ID)) ne 0

                      '' --these numbers will not allow rollback
                      '' if rollback desired, remove abs.f
  DISPLACEMENT = abs.f(normal.f(0,.25, 5))

  if X.M < X.ID         ''displace left
    NEW.X.M = X.ID - (PLOW.WID + DISPLACEMENT)
   else                   ''displace right
    NEW.X.M = X.ID + (PLOW.WID + DISPLACEMENT)
  endif

  NEW.Y.M = Y.M

  MINE.X(M(MINE.ID)) = NEW.X.M
  MINE.Y(M(MINE.ID)) = NEW.Y.M

  if GRAPH.ON = 1
    erase M(MINE.ID)
    select case MINE.TYPE(M(MINE.ID))
      case 1
        show M(MINE.ID) with "rmine.icn"
      case 2
        show M(MINE.ID) with "bbmine.icn"
      case 3
        show M(MINE.ID) with "blmine.icn"
    endselect

    X = MINE.X(M(MINE.ID))
    Y = MINE.Y(M(MINE.ID))
    let location.a(M(MINE.ID)) = location.f(X,Y)
  endif
endif

return
end ''routine FULL.PLOW
```

```
routine GET.DATA

define DEVPTR as a pointer variable

define MASTER.FORM as a pointer variable
define INPUT.FORM as a pointer variable
define PK.FORM as a pointer variable


define TF.FORM as a pointer variable      ''pointer to taskforce menu

define PLOW1.FORM as a pointer variable   ''pointer to plow1 (fw) menu
define PLOW2.FORM as a pointer variable   ''pointer to plow2 (tw) menu
define FWPLOW.FORM as a pointer variable  ''pointer to fwplow menu
define TWPLOW.FORM as a pointer variable  ''pointer to twplow menu

define TRACK1.FORM as a pointer variable  ''pointer to track 1 menu
define TRACK2.FORM as a pointer variable  ''pointer to track 2 menu
define TRACK3.FORM as a pointer variable  ''pointer to track 3 menu

define MINE.DAT.FORM as a pointer variable

defina DISTANCE as a real variable
defina FIELD.ID as a text variable
defina FIELD.ID.2 as a taxt variabla

open unit 2 for output,
  file name is "OUTFILE"
use 2 for output

call DEVINIT.R("VT,GRAPHIC") yielding DEVPTR
open 7 for input, device = DEVPTR
open 8 for output, device = DEVPTR
use 8 for graphic output

''Default values for taskforce input form
''=============================================

show TF.FORM with "tf.frm"

let ddval.a(dfield.f("FULL.PLOW", TF.FORM)) = F.PLOWS
let ddval.a(dfield.f("TRACK.PLOW", TF.FORM)) = T.PLOWS
let ddval.a(dfield.f("TRACK1", TF.FORM)) = TRACK1
let ddval.a(dfield.f("TRACK2", TF.FORM)) = TRACK2
let ddval.a(dfield.f("TRACK3", TF.FORM)) = TRACK3
''---------------------------------------------


''Default values for vehicle type 1 (full width plow)
''=================================================

show PLOW1.FORM with "plow1.frm"

let ddval.a(dfield.f("NAV.ERROR", PLOW1.FORM)) = NAV.ERR.1
let ddval.a(dfield.f("TANK.WIDTH", PLOW1.FORM)) = E.WIDTH.1
let ddval.a(dfield.f("TANK.LENGTH", PLOW1.FORM)) = E.LENGTH.1
let ddval.a(dfield.f("TRACK.WIDTH", PLOW1.FORM)) = E.TRACK.WIDTH.1
''---------------------------------------------

''Default values for full width plow specification input form
''=============================================


```

```
  63  show FWPLOW.FORM with "fwplow.frm"
  64
  65  let ddval.a(dfield.f("PLOW.EFFECTIVENESS", FWPLOW.FORM)) = PLOW.EFFECT.1
  66  let ddval.a(dfield.f("PLOW.WIDTH", FWPLOW.FORM)) = PLOW.WIDTH.1
  67  let ddval.a(dfield.f("PLOW.SPEED", FWPLOW.FORM)) = E.SPEED.1
  68  ''-----------------------------------------------
  69
  70
  71
  72
  73
  74  ''Default values for vehicle type 2 (track-width plow)
  75  ''=====================================================
  76
  77  show PLOW2.FORM with "plow2.frm"
  78
  79  let ddval.a(dfield.f("NAV.ERROR", PLOW2.FORM)) = NAV.ERR.2
  80  let ddval.a(dfield.f("TANK.WIDTH", PLOW2.FORM)) = E.WIDTH.2
  81  let ddval.a(dfield.f("TANK.LENGTH", PLOW2.FORM)) = E.LENGTH.2
  82  let ddval.a(dfield.f("TRACK.WIDTH", PLOW2.FORM)) = E.TRACK.WIDTH.2
  83  ''-----------------------------------------------
  84
  85  ''Default values for track width plow specification input form
  86  ''============================================================
  87
  88  show TWPLOW.FORM with "twplow.frm"
  89
  90  let ddval.a(dfield.f("PLOW.EFFECTIVENESS", TWPLOW.FORM)) = PLOW.EFFECT.2
  91  let ddval.a(dfield.f("PLOW.WIDTH", TWPLOW.FORM)) = PLOW.WIDTH.2
  92  let ddval.a(dfield.f("PLOW.SPEED", TWPLOW.FORM)) = E.SPEED.2
  93  let ddval.a(dfield.f("EFF.WIDTH", TWPLOW.FORM)) = EFF.WIDTH
  94  ''-----------------------------------------------
  95
  96
  97  ''Default values for vehicle type 3 (track type 1)
  98  ''=================================================
  99
 100  show TRACK1.FORM with "track1.frm"
 101
 102  let ddval.a(dfield.f("NAV.ERROR", TRACK1.FORM)) = NAV.ERR.3
 103  let ddval.a(dfield.f("TANK.WIDTH", TRACK1.FORM)) = E.WIDTH.3
 104  let ddval.a(dfield.f("TANK.LENGTH", TRACK1.FORM)) = E.LENGTH.3
 105  let ddval.a(dfield.f("TRACK.WIDTH", TRACK1.FORM)) = E.TRACK.WIDTH.3
 106  ''-----------------------------------------------
 107
 108
 109  ''Default values for vehicle type 4 (track type 2)
 110  ''=================================================
 111
 112  show TRACK2.FORM with "track2.frm"
 113
 114  let ddval.a(dfield.f("NAV.ERROR", TRACK2.FORM)) = NAV.ERR.4
 115  let ddval.a(dfield.f("TANK.WIDTH", TRACK2.FORM)) = E.WIDTH.4
 116  let ddval.a(dfield.f("TANK.LENGTH", TRACK2.FORM)) = E.LENGTH.4
 117  let ddval.a(dfield.f("TRACK.WIDTH", TRACK2.FORM)) = E.TRACK.WIDTH.4
 118  ''-----------------------------------------------
 119
 120
 121  ''Default values for vehicle type 5 (track type 3)
 122  ''=================================================
 123
 124  show TRACK3.FORM with "track3.frm"
```

```
125
126  let ddval.a(dfield.f("NAV.ERROR", TRACK3.FORM)) = NAV.ERR.5
127  let ddval.a(dfield.f("TANK.WIDTH", TRACK3.FORM)) = E.WIDTH.5
128  let ddval.a(dfield.f("TANK.LENGTH", TRACK3.FORM)) = E.LENGTH.5
129  let ddval.a(dfield.f("TRACK.WIDTH", TRACK3.FORM)) = E.TRACK.WIDTH.5
130  ''-----------------------------------------------
131
132
133
134  ''Default values for minefield data form
135  ''===============================================
136
137  show MINE.DAT.FORM with "minedat.frm"
138
139  let ddval.a(dfield.f("P.MINES", MINE.DAT.FORM)) = P.MINES
140  let ddval.a(dfield.f("I.MINES", MINE.DAT.FORM)) = I.MINES
141  let ddval.a(dfield.f("C.MINES", MINE.DAT.FORM)) = C.MINES
142  let ddval.a(dfield.f("T.MINES", MINE.DAT.FORM)) = T.MINES
143  let ddval.a(dfield.f("W.MINES", MINE.DAT.FORM)) = W.MINES
144  ''-----------------------------------------------
145
146
147
148  ''Default values for mine specification input form
149  ''===============================================
150
151  show PK.FORM with "pkform.frm"
152
153  let ddval.a(dfield.f("P.RADIUS", PK.FORM)) = P.RADIUS
154  let ddval.a(dfield.f("I.RADIUS", PK.FORM)) = I.RADIUS
155  let ddval.a(dfield.f("C.RADIUS", PK.FORM)) = C.RADIUS
156  let ddval.a(dfield.f("T.RADIUS", PK.FORM)) = T.RADIUS
157  let ddval.a(dfield.f("W.RADIUS", PK.FORM)) = W.RADIUS
158
159  let ddval.a(dfield.f("ONE.ONE", PK.FORM))     = MINE.PK(1,1)
160  let ddval.a(dfield.f("ONE.TWO", PK.FORM))     = MINE.PK(1,2)
161  let ddval.a(dfield.f("ONE.THREE", PK.FORM))   = MINE.PK(1,3)
162  let ddval.a(dfield.f("ONE.FOUR", PK.FORM))    = MINE.PK(1,4)
163  let ddval.a(dfield.f("ONE.FIVE", PK.FORM))    = MINE.PK(1,5)
164
165  let ddval.a(dfield.f("TWO.ONE", PK.FORM))     = MINE.PK(2,1)
166  let ddval.a(dfield.f("TWO.TWO", PK.FORM))     = MINE.PK(2,2)
167  let ddval.a(dfield.f("TWO.THREE", PK.FORM))   = MINE.PK(2,3)
168  let ddval.a(dfield.f("TWO.FOUR", PK.FORM))    = MINE.PK(2,4)
169  let ddval.a(dfield.f("TWO.FIVE", PK.FORM))    = MINE.PK(2,5)
170
171  let ddval.a(dfield.f("THREE.ONE", PK.FORM))   = MINE.PK(3,1)
172  let ddval.a(dfield.f("THREE.TWO", PK.FORM))   = MINE.PK(3,2)
173  let ddval.a(dfield.f("THREE.THREE", PK.FORM)) = MINE.PK(3,3)
174  let ddval.a(dfield.f("THREE.FOUR", PK.FORM))  = MINE.PK(3,4)
175  let ddval.a(dfield.f("THREE.FIVE", PK.FORM))  = MINE.PK(3,5)
176
177  let ddval.a(dfield.f("FOUR.ONE", PK.FORM))    = MINE.PK(4,1)
178  let ddval.a(dfield.f("FOUR.TWO", PK.FORM))    = MINE.PK(4,2)
179  let ddval.a(dfield.f("FOUR.THREE", PK.FORM))  = MINE.PK(4,3)
180  let ddval.a(dfield.f("FOUR.FOUR", PK.FORM))   = MINE.PK(4,4)
181  let ddval.a(dfield.f("FOUR.FIVE", PK.FORM))   = MINE.PK(4,5)
182
183  let ddval.a(dfield.f("FIVE.ONE", PK.FORM))    = MINE.PK(5,1)
184  let ddval.a(dfield.f("FIVE.TWO", PK.FORM))    = MINE.PK(5,2)
185  let ddval.a(dfield.f("FIVE.THREE", PK.FORM))  = MINE.PK(5,3)
186  let ddval.a(dfield.f("FIVE.FOUR", PK.FORM))   = MINE.PK(5,4)
```

```
187  let ddval.a(dfield.f("FIVE.FIVE", PK.FORM))   = MINE.PK(5,5)
188
189
190  '' initialize pointer to master input form
191  ''-------------------------------------------------------
192  show MASTER.FORM with "master.frm"
193
194
195
196
197
198
199  ''-------------------------------------------------------------
200
201  ''                    DATA ENTRY
202
203  ''=============================================================
204
205
206
207  'MASTER'
208
209  let FIELD.ID.2 = accept.f(MASTER.FORM,0)
210
211  select case FIELD.ID.2
212
213    case "MODEL.PARAM"
214     ''model parameter input screen
215     go to 'MASTER'
216
217    case "SCEN.RUN.PARAM"
218     ''scenario/run parameters input screen
219     go to 'MASTER'
220
221    case "TASKFORCE.DATA"
222
223      'TASKFORCE'
224      let FIELD.ID = accept.f(TF.FORM,0)
225
226      let F.PLOWS = ddval.a(dfield.f("FULL.PLOW", TF.FORM))
227      let T.PLOWS = ddval.a(dfield.f("TRACK.PLOW", TF.FORM))
228      let TRACK1  = ddval.a(dfield.f("TRACK1", TF.FORM))
229      let TRACK2  = ddval.a(dfield.f("TRACK2", TF.FORM))
230      let TRACK3  = ddval.a(dfield.f("TRACK3", TF.FORM))
231
232
233      select case FIELD.ID
234        case "ED.F.PLOW"
235
236          let FIELD.ID = accept.f(PLOW1.FORM,0)
237
238          let NAV.ERR.1     = ddval.a(dfield.f("NAV.ERROR", PLOW1.FORM))
239          let E.WIDTH.1     = ddval.a(dfield.f("TANK.WIDTH", PLOW1.FORM))
240          let E.LENGTH.1    = ddval.a(dfield.f("TANK.LENGTH", PLOW1.FORM))
241          let E.TRACK.WIDTH.1 = ddval.a(dfield.f("TRACK.WIDTH", PLOW1.FORM))
242
243          let FIELD.ID = accept.f(FWPLOW.FORM,0)
244
245          let PLOW.EFFECT.1 = ddval.a(dfield.f("PLOW.EFFECTIVENESS", FWPLOW.FORM))
246          let PLOW.WIDTH.1 = ddval.a(dfield.f("PLOW.WIDTH", FWPLOW.FORM))
247          let E.SPEED.1 = ddval.a(dfield.f("PLOW.SPEED", FWPLOW.FORM))
248
```

```
249          go to 'TASKFORCE'
250
251        case "ED.T.PLOW"
252
253          let FIELD.ID = accept.f(PLOW2.FORM,0)
254
255          let NAV.ERR.2     = ddval.a(dfield.f("NAV.ERROR", PLOW2.FORM))
256          let E.WIDTH.2     = ddval.a(dfield.f("TANK.WIDTH", PLOW2.FORM))
257          let E.LENGTH.2    = ddval.a(dfield.f("TANK.LENGTH", PLOW2.FORM))
258          let E.TRACK.WIDTH.2 = ddval.a(dfield.f("TRACK.WIDTH", PLOW2.FORM))
259
260          let FIELD.ID = accept.f(TWPLOW.FORM,0)
261
262          let PLOW.EFFECT.2 = ddval.a(dfield.f("PLOW.EFFECTIVENESS", TWPLOW.FORM))
263          let PLOW.WIDTH.2 = ddval.a(dfield.f("PLOW.WIDTH", TWPLOW.FORM))
264          let E.SPEED.2 = ddval.a(dfield.f("PLOW.SPEED", TWPLOW.FORM))
265          let EFF.WIDTH = ddval.a(dfield.f("EFF.WIDTH", TWPLOW.FORM))
266
267          go to 'TASKFORCE'
268
269        case "ED.T1"
270          let FIELD.ID = accept.f(TRACK1.FORM,0)
271
272          let NAV.ERR.3     = ddval.a(dfield.f("NAV.ERROR", TRACK1.FORM))
273          let E.WIDTH.3     = ddval.a(dfield.f("TANK.WIDTH", TRACK1.FORM))
274          let E.LENGTH.3    = ddval.a(dfield.f("TANK.LENGTH", TRACK1.FORM))
275          let E.TRACK.WIDTH.3 = ddval.a(dfield.f("TRACK.WIDTH", TRACK1.FORM))
276
277          go to 'TASKFORCE'
278
279        case "ED.T2"
280          let FIELD.ID = accept.f(TRACK2.FORM,0)
281
282          let NAV.ERR.4     = ddval.a(dfield.f("NAV.ERROR", TRACK2.FORM))
283          let E.WIDTH.4     = ddval.a(dfield.f("TANK.WIDTH", TRACK2.FORM))
284          let E.LENGTH.4    = ddval.a(dfield.f("TANK.LENGTH", TRACK2.FORM))
285          let E.TRACK.WIDTH.4 = ddval.a(dfield.f("TRACK.WIDTH", TRACK2.FORM))
286
287          go to 'TASKFORCE'
288
289        case "ED.T3"
290          let FIELD.ID = accept.f(TRACK3.FORM,0)
291
292          let NAV.ERR.5     = ddval.a(dfield.f("NAV.ERROR", TRACK3.FORM))
293          let E.WIDTH.5     = ddval.a(dfield.f("TANK.WIDTH", TRACK3.FORM))
294          let E.LENGTH.5    = ddval.a(dfield.f("TANK.LENGTH", TRACK3.FORM))
295          let E.TRACK.WIDTH.5 = ddval.a(dfield.f("TRACK.WIDTH", TRACK3.FORM))
296
297          go to 'TASKFORCE'
298
299       case "TF.RETURN"
300
301          go to 'MASTER'
302
303       endselect   ''field.id
304
305     case "MINEFIELD.DATA"
306
307       'MINEFIELD'
308
309       let FIELD.ID = accept.f(MINE.DAT.FORM,0)
310
```

```
311       let P.MINES = ddval.a(dfield.f("P.MINES", MINE.DAT.FORM))
312       let I.MINES = ddval.a(dfield.f("I.MINES", MINE.DAT.FORM))
313       let C.MINES = ddval.a(dfield.f("C.MINES", MINE.DAT.FORM))
314       let T.MINES = ddval.a(dfield.f("T.MINES", MINE.DAT.FORM))
315       let W.MINES = ddval.a(dfield.f("W.MINES", MINE.DAT.FORM))
316
317       if FIELD.ID = "M.EDIT"
318
319         let FIELD.ID = accept.f(PK.FORM,0)
320
321         let P.RADIUS  = ddval.a(dfield.f("P.RADIUS", PK.FORM))
322         let I.RADIUS  = ddval.a(dfield.f("I.RADIUS", PK.FORM))
323         let C.RADIUS  = ddval.a(dfield.f("C.RADIUS", PK.FORM))
324         let T.RADIUS  = ddval.a(dfield.f("T.RADIUS", PK.FORM))
325         let W.RADIUS  = ddval.a(dfield.f("W.RADIUS", PK.FORM))
326
327         let MINE.PK(1,1)  = ddval.a(dfield.f("ONE.ONE", PK.FORM))
328         let MINE.PK(1,2)  = ddval.a(dfield.f("ONE.TWO", PK.FORM))
329         let MINE.PK(1,3)  = ddval.a(dfield.f("ONE.THREE", PK.FORM))
330         let MINE.PK(1,4)  = ddval.a(dfield.f("ONE.FOUR", PK.FORM))
331         let MINE.PK(1,5)  = ddval.a(dfield.f("ONE.FIVE", PK.FORM))
332
333         let MINE.PK(2,1)  = ddval.a(dfield.f("TWO.ONE", PK.FORM))
334         let MINE.PK(2,2)  = ddval.a(dfield.f("TWO.TWO", PK.FORM))
335         let MINE.PK(2,3)  = ddval.a(dfield.f("TWO.THREE", PK.FORM))
336         let MINE.PK(2,4)  = ddval.a(dfield.f("TWO.FOUR", PK.FORM))
337         let MINE.PK(2,5)  = ddval.a(dfield.f("TWO.FIVE", PK.FORM))
338
339         let MINE.PK(3,1)  = ddval.a(dfield.f("THREE.ONE", PK.FORM))
340         let MINE.PK(3,2)  = ddval.a(dfield.f("THREE.TWO", PK.FORM))
341         let MINE.PK(3,3)  = ddval.a(dfield.f("THREE.THREE", PK.FORM))
342         let MINE.PK(3,4)  = ddval.a(dfield.f("THREE.FOUR", PK.FORM))
343         let MINE.PK(3,5)  = ddval.a(dfield.f("THREE.FIVE", PK.FORM))
344
345         let MINE.PK(4,1)  = ddval.a(dfield.f("FOUR.ONE", PK.FORM))
346         let MINE.PK(4,2)  = ddval.a(dfield.f("FOUR.TWO", PK.FORM))
347         let MINE.PK(4,3)  = ddval.a(dfield.f("FOUR.THREE", PK.FORM))
348         let MINE.PK(4,4)  = ddval.a(dfield.f("FOUR.FOUR", PK.FORM))
349         let MINE.PK(4,5)  = ddval.a(dfield.f("FOUR.FIVE", PK.FORM))
350
351         let MINE.PK(5,1)  = ddval.a(dfield.f("FIVE.ONE", PK.FORM))
352         let MINE.PK(5,2)  = ddval.a(dfield.f("FIVE.TWO", PK.FORM))
353         let MINE.PK(5,3)  = ddval.a(dfield.f("FIVE.THREE", PK.FORM))
354         let MINE.PK(5,4)  = ddval.a(dfield.f("FIVE.FOUR", PK.FORM))
355         let MINE.PK(5,5)  = ddval.a(dfield.f("FIVE.FIVE", PK.FORM))
356
357       endif
358
359       if FIELD.ID = "RETURN.TO.MFDATA"
360         go to 'MASTER'
361       endif
362
363
364     case  "OVERWATCH.DATA"
365
366       go to 'MASTER'
367       ''display overwatch.data input form
368
369     case "OUTPUT.OPTIONS"
370
371       go to 'MASTER'
372       ''display output options input form
```

```
373
374
375    case "START.MODEL"
376
377
378  endselect ''field.id.2
379
380
381  show INPUT.FORM with "input.frm"
382
383  let ddval.a(dfield.f("MINEFIELD.DEPTH", INPUT.FORM)) = 25.0
384  let ddval.a(dfield.f("CP.INTERVAL", INPUT.FORM)) = 20.0
385  let ddval.a(dfield.f("DISTANCE", INPUT.FORM)) = 160.0
386  let ddval.a(dfield.f("INTERVAL", INPUT.FORM)) = 30.0
387  let ddval.a(dfield.f("E.SPEED", INPUT.FORM)) = 8.0
388  let ddval.a(dfield.f("GRAPH.ON", INPUT.FORM)) = 1
389  let ddval.a(dfield.f("NUMBER.OF.RUNS", INPUT.FORM)) = 1
390  let ddval.a(dfield.f("PLOW.INTERVAL", INPUT.FORM)) = 50.0
391
392  let FIELD.ID = accept.f(INPUT.FORM,0)
393
394  let CP.INTERVAL = ddval.a(dfield.f("CP.INTERVAL", INPUT.FORM))
395  let DISTANCE    = ddval.a(dfield.f("DISTANCE", INPUT.FORM))
396  let INTERVAL    = ddval.a(dfield.f("INTERVAL", INPUT.FORM))
397  let E.SPEED     = ddval.a(dfield.f("E.SPEED", INPUT.FORM))
398  let GRAPH.ON    = ddval.a(dfield.f("GRAPH.ON", INPUT.FORM))
399  let NUMBER.OF.RUNS = ddval.a(dfield.f("NUMBER.OF.RUNS", INPUT.FORM))
400  let MINEFIELD.DEPTH = ddval.a(dfield.f("MINEFIELD.DEPTH", INPUT.FORM))
401  let PLOW.INTERVAL = ddval.a(dfield.f("PLOW.INTERVAL", INPUT.FORM))
402
403
404  NUM.ELEMENT = F.PLOWS + T.PLOWS + TRACK1 + TRACK2 + TRACK3
405  NUM.MINE = P.MINES + I.MINES + C.MINES + T.MINES + W.MINES
406
407  NUM.CP = (DISTANCE / 20.0) + 1.0
408
409  return
410  end ''GET.DATA
```

```
    1  event GRAPHIC.UPDATE
    2
    3  ''THIS EVENT IS USED TO UPDATE THE LOCATION OF THE GRAPHIC ENTITIES AND
    4  ''ANY SPECIFIC RATE -- CURRENTLY HARD CODED, BUT CAN BE MADE INTO A
    5  ''USER SELECTED REFREASH RATE
    6
    7  define I as an integer variable
    8
    9  for I = 1 to NUM.ELEMENT do
   10    if (SPEED(E(I)) > 0)
   11      schedule a graphic.update at time.v + .025
   12      leave
   13    endif
   14  loop
   15
   16  call UPDATE.LOCATION
   17
   18  return
   19  end ''GRAPHIC.UPDATE
```

```
routine INITIALIZE

''THIS ROUTINE IS USED TO CREATE ALL ELEMENTS. IF APPROPRIATE, IT CAUSES A
''MOVEMENT.PLAN TO BE INDIVIDUALLY CREATED FOR EACH ONE. REINITIALIZES ALL
''DATA RUN COUNTERS AND STARTS THE SIMULATION.

define I as an integer variable
define J as an integer variable
define DELAY as a double variable
define PLOW.DELAY as a double variable
define INCREMENT as a real variable
define COUNTER1 as an integer variable
define COUNTER2 as an integer variable  ''used to accurately run through
                                        ''element array

NUM.OBSTACLE = NUM.ELEMENT

reserve E as NUM.ELEMENT
reserve OB as NUM.OBSTACLE
reserve RO as NUM.RED.OVERWATCH
reserve SCH.EVENTS(*) as NUM.ELEMENT
reserve EVENT.LIST(*) as NUM.ELEMENT
reserve BYPASS.MAP(*,*,*) as NUM.ELEMENT by 4 by 5
reserve MOVEMENT.PLAN(*,*,*) as NUM.ELEMENT by NUM.CP by 5
reserve CP(*) as NUM.CP
reserve TIME.OF.DEATH(*) as NUM.ELEMENT
reserve TIME.OF.COMPLETION as NUM.ELEMENT

                    ''if multiple runs, turn off graphics,
                    ''and initialize data variables each run
if (NUMBER.OF.RUNS > 1)

  for J = 1 to NUM.ELEMENT
    do
    TIME.OF.DEATH(J) = 0
    TIME.OF.COMPLETION(J) = 0
  loop

  MINE.KILLS = 0
  MINE.KILL.1 = 0
  MINE.KILL.2 = 0
  MINE.KILL.3 = 0
  MINE.KILL.4 = 0
  MINE.KILL.5 = 0

  MINE.MOVED = 0

  R.O.SHOTS = 0
  R.O.KILL = 0

  TOTAL.LOST = 0
  LOSS.RATE = 0
  LOST.TO.MINES.BYPASSING = 0
  LOST.TO.FIRES.BYPASSING = 0

  time.v = 0.0

endif

for I = 1 to NUM.ELEMENT
  do
  create a ELEMENT called E(I)
```

```
   create a OBSTACLE called OB(I)
loop

for I = 1 to NUM.RED.OVERWATCH
  do
  create a RED.OVERWATCH called RO(I)
loop

INCREMENT = -10.0

for I = 1 to NUM.RED.OVERWATCH
  do

  R.O.X(RO(I)) = INCREMENT
  INCREMENT = INCREMENT + 10.0
  R.O.STATUS(RO(I)) = 1
  R.O.Y(RO(I)) = NUM.CP * CP.INTERVAL
  R.O.BETA(RO(I)) = R.O.BETA.1
  R.O.RATE(RO(I)) = R.O.RATE.1
  R.O.TYPE(RO(I)) = 1
  R.O.NUM(RO(I)) = I
loop

COUNTER1 = 1
if F.PLOWS ne 0

  COUNTER1 = 1
  COUNTER2 = F.PLOWS

  for I = COUNTER1 to COUNTER2
    do
    ''initialize full plow elements
    ELEMENT.NUM(E(I)) = I
    ELEMENT.TYPE(E(I)) = 1
    PLOW.WIDTH(E(I)) = PLOW.WIDTH.1
    PLOW.EFFECT(E(I)) = PLOW.EFFECT.1
    P.STATUS(E(I)) = 1

    call MAKE.ROUTE giving I
                              ''initialize element start point location
                              ''based on movement plan
    ELEMENT.X(E(I)) = MOVEMENT.PLAN(I,1,1)
    ELEMENT.Y(E(I)) = MOVEMENT.PLAN(I,1,2)
    STATUS(E(I)) = 1
    WIDTH(E(I)) = E.WIDTH.1
    LENGTH(E(I)) = E.LENGTH.1
    ELEM.RADIUS(E(I)) = sqrt.f((WIDTH(E(I)) * LENGTH(E(I)))/Pi.c)
    TRACK.WIDTH(E(I)) = E.TRACK.WIDTH.1
  loop
endif


if T.PLOWS ne 0
  COUNTER1 = COUNTER1 + COUNTER2
  COUNTER2 = COUNTER2 + T.PLOWS

  for I = COUNTER1 to COUNTER2
    do
    ''initialize track plow elements
    ELEMENT.NUM(E(I)) = I
    ELEMENT.TYPE(E(I)) = 2
    PLOW.WIDTH = PLOW.WIDTH.2
```

```
125      PLOW.EFFECT = PLOW.EFFECT.2
126
127      call MAKE.ROUTE giving I
128                                ''initialize element start point location
129                                ''based on movement plan
130      ELEMENT.X(E(I)) = MOVEMENT.PLAN(I,1,1)
131      ELEMENT.Y(E(I)) = MOVEMENT.PLAN(I,1,2)
132      STATUS(E(I)) = 1
133      WIDTH(E(I)) = E.WIDTH.2
134      LENGTH(E(I)) = E.LENGTH.2
135      ELEM.RADIUS(E(I)) = sqrt.f((WIDTH(E(I)) * LENGTH(E(I)))/Pi.c)
136      TRACK.WIDTH(E(I)) = E.TRACK.WIDTH.2
137    loop
138  endif
139
140  if TRACK1 ne 0
141
142    COUNTER1 = COUNTER1 + COUNTER2
143    COUNTER2 = COUNTER2 + TRACK1
144
145    for I = COUNTER1 to COUNTER2
146      do
147      ''initialize track type 1 elements
148      ELEMENT.NUM(E(I)) = I
149      ELEMENT.TYPE(E(I)) = 3
150      call MAKE.ROUTE giving I
151                                ''initialize element start point location
152                                ''based on movement plan
153      ELEMENT.X(E(I)) = MOVEMENT.PLAN(I,1,1)
154      ELEMENT.Y(E(I)) = MOVEMENT.PLAN(I,1,2)
155      STATUS(E(I)) = 1
156      WIDTH(E(I)) = E.WIDTH.3
157      LENGTH(E(I)) = E.LENGTH.3
158      ELEM.RADIUS(E(I)) = sqrt.f((WIDTH(E(I)) * LENGTH(E(I)))/Pi.c)
159      TRACK.WIDTH(E(I)) = E.TRACK.WIDTH.3
160    loop
161  endif
162
163  if TRACK2 ne 0
164
165    COUNTER1 = COUNTER1 + COUNTER2
166    COUNTER2 = COUNTER2 + TRACK2
167
168    for I = COUNTER1 to COUNTER2
169      do
170      ''initialize track type 2 elements
171      ELEMENT.NUM(E(I)) = I
172      ELEMENT.TYPE(E(I)) = 4
173      call MAKE.ROUTE giving I
174                                ''initialize element start point location
175                                ''based on movement plan
176      ELEMENT.X(E(I)) = MOVEMENT.PLAN(I,1,1)
177      ELEMENT.Y(E(I)) = MOVEMENT.PLAN(I,1,2)
178      STATUS(E(I)) = 1
179      WIDTH(E(I)) = E.WIDTH.4
180      LENGTH(E(I)) = E.LENGTH.4
181      ELEM.RADIUS(E(I)) = sqrt.f((WIDTH(E(I)) * LENGTH(E(I)))/Pi.c)
182      TRACK.WIDTH(E(I)) = E.TRACK.WIDTH.4
183    loop
184  endif
185
186  if TRACK3 ne 0
```

```
187
188    COUNTER1 = COUNTER1 + COUNTER2
189    COUNTER2 = COUNTER2 + TRACK3
190
191    for I = COUNTER1 to COUNTER2
192      do
193      ''initialize track type 3 elements
194      ELEMENT.NUM(E(I)) = I
195      ELEMENT.TYPE(E(I)) = 5
196      call MAKE.ROUTE giving I
197
198                              ''initialize element start point location
199                              ''based on movement plan
200      ELEMENT.X(E(I)) = MOVEMENT.PLAN(I,1,1)
201      ELEMENT.Y(E(I)) = MOVEMENT.PLAN(I,1,2)
202      STATUS(E(I)) = 1
203      WIDTH(E(I)) = E.WIDTH.5
204      LENGTH(E(I)) = E.LENGTH.5
205      ELEM.RADIUS(E(I)) = sqrt.f((WIDTH(E(I)) * LENGTH(E(I)))/Pi.c)
206      TRACK.WIDTH(E(I)) = E.TRACK.WIDTH.5
207    loop
208  endif
209
210  call MAKE.MINEFIELD
211
212  for I = 1 to F.PLOWS
213    do
214    DELAY = INTERVAL/(E.SPEED * (1000.0/60.0))
215    DELAY = (I - 1) * DELAY
216    schedule a START giving I at (time.v + DELAY)
217  loop
218
219  PLOW.DELAY = PLOW.INTERVAL / (E.SPEED * (1000.0 / 60.0))
220
221  for I = (F.PLOWS + 1) to NUM.ELEMENT
222    do
223                          ''compute time it will take for elements to
224                          ''cover interval distance at given speed
225    DELAY = INTERVAL / (E.SPEED * (1000.0 / 60.0))
226                          ''convert delay into actual delay based on
227                          ''position in line of vehicles
228    DELAY = PLOW.DELAY + ((I - 1) * DELAY)
229
230    schedule a START giving I at (time.v + DELAY)
231  loop
232
233  for I = 1 to NUM.RED.OVERWATCH
234    do
235    call R.DIRECT.OVERWATCH giving I
236  loop
237
238  if GRAPH.ON = 1
239    schedule a GRAPHIC.UPDATE at (time.v + .05)
240  endif
241
242  start simulation
243  return
244  end ''INITIALIZE
```

```
routine MAKE.HEX given E.ID, O.ID

''GIVEN AN ELEMENT ID AND AN OBSTACLE ID, THIS ROUTINE CALCULATES AND STORES
''THE COORDINATES AND EQUATIONS FOR AN EQUALATERAL HEXAGON TO BE USED AS
''THE CENTERLINE OF THE ELEMENTS BYPASS PATH AROUND THE OBSTACLE

define E.ID as an integer variable
define O.ID as an integer variable
define O.X as a real variable
define O.Y as a real variable
define O.RADIUS as a real variable
define E.RADIUS as a real variable
define RADIUS as a real variable
define COEF1 as a real variable
define COEF2 as a real variable

O.X = OBSTACLE.X(OB(O.ID))
O.Y = OBSTACLE.Y(OB(O.ID))
O.RADIUS = OBS.RADIUS(OB(O.ID))

E.RADIUS = ELEM.RADIUS(E(E.ID))

RADIUS = O.RADIUS + E.RADIUS

                '' coeffecients derived from 30-60-90 degree triangle
COEF1 = (1.0/sqrt.f(3.0)) * RADIUS
COEF2 = (2.0/sqrt.f(3.0)) * RADIUS

PT1.X(OB(O.ID)) = O.X
PT1.Y(OB(O.ID)) = O.Y - COEF2
PT2.X(OB(O.ID)) = O.X + RADIUS
PT2.Y(OB(O.ID)) = O.Y - COEF1
PT3.X(OB(O.ID)) = O.X + RADIUS
PT3.Y(OB(O.ID)) = O.Y + COEF1
PT4.X(OB(O.ID)) = O.X
PT4.Y(OB(O.ID)) = O.Y + COEF2
PT5.X(OB(O.ID)) = O.X - RADIUS
PT5.Y(OB(O.ID)) = O.Y + COEF1
PT6.X(OB(O.ID)) = O.X - RADIUS
PT6.Y(OB(O.ID)) = O.Y - COEF1

SLOPE.A(OB(O.ID)) =  (PT2.Y(OB(O.ID)) - PT1.Y(OB(O.ID))) /
                     (PT2.X(OB(O.ID)) - PT1.X(OB(O.ID)))

SLOPE.F(OB(O.ID)) =  (PT1.Y(OB(O.ID)) - PT6.Y(OB(O.ID))) /
                     (PT1.X(OB(O.ID)) - PT6.X(OB(O.ID)))

SLOPE.C(OB(O.ID)) =  (PT3.Y(OB(O.ID)) - PT4.Y(OB(O.ID))) /
                     (PT3.X(OB(O.ID)) - PT4.X(OB(O.ID)))

SLOPE.D(OB(O.ID)) =  (PT4.Y(OB(O.ID)) - PT5.Y(OB(O.ID))) /
                     (PT4.X(OB(O.ID)) - PT5.X(OB(O.ID)))


Y.INT.A(OB(O.ID)) =  PT1.Y(OB(O.ID)) - SLOPE.A(OB(O.ID)) * PT1.X(OB(O.ID))
Y.INT.F(OB(O.ID)) =  PT1.Y(OB(O.ID)) - SLOPE.F(OB(O.ID)) * PT1.X(OB(O.ID))
Y.INT.C(OB(O.ID)) =  PT4.Y(OB(O.ID)) - SLOPE.C(OB(O.ID)) * PT4.X(OB(O.ID))
Y.INT.D(OB(O.ID)) =  PT4.Y(OB(O.ID)) - SLOPE.D(OB(O.ID)) * PT4.X(OB(O.ID))


return
```

```
   63
   64  end ''MAKE.HEX
```

```
routine MAKE.MINEFIELD

''ROUTINE TO GENERATE RANDOM, UNIFORM DENSITY MINEFIELD

define I as an integer variable
define X, Y as real variables
define DEPTH as a real variable
define MF.WIDTH as a real variable
define MF.DEPTH as a real variable
define MF.Y as a real variable
define MF.B, MF.T as real variables
define AREA as a real variable
define WIDTH as a real variable

reserve M as NUM.MINE
                                ''depth of display
DEPTH = NUM.CP * CP.INTERVAL
                                ''center the screen
MF.WIDTH = DEPTH/2.0

                                ''center the mines on screen
MF.DEPTH = MINEFIELD.DEPTH / 2.0

MF.Y = .5 * DEPTH
MF.B = MF.Y - MF.DEPTH
MF.T = MF.Y + MF.DEPTH
AREA = (2.0 * MF.WIDTH) * (2.0 * MF.DEPTH)
WIDTH = 2.0 * MF.WIDTH

AREA.DENSITY = NUM.MINE/AREA
AREA.DENSITY.TYPE.1 = P.MINES/AREA
AREA.DENSITY.TYPE.2 = I.MINES/AREA
AREA.DENSITY.TYPE.3 = C.MINES/AREA
AREA.DENSITY.TYPE.4 = T.MINES/AREA
AREA.DENSITY.TYPE.5 = W.MINES/AREA

LINEAR.DENSITY = NUM.MINE/WIDTH
LINEAR.DENSITY.TYPE.1 = P.MINES/WIDTH
LINEAR.DENSITY.TYPE.2 = I.MINES/WIDTH
LINEAR.DENSITY.TYPE.3 = C.MINES/WIDTH
LINEAR.DENSITY.TYPE.4 = T.MINES/WIDTH
LINEAR.DENSITY.TYPE.5 = W.MINES/WIDTH

if P.MINES ne 0
  for I = 1 to P.MINES
    do

    create MINE called M(I)
    MINE.NUM(M(I)) = I
    X = UNIFORM.F(-MF.WIDTH,MF.WIDTH,2)
    MINE.X(M(I)) = X
    Y = UNIFORM.F(MF.B, MF.T, 2)
    MINE.Y(M(I)) = Y
    RADIUS(M(I)) = P.RADIUS
    MINE.TYPE(M(I)) = 1
    MINE.STATUS(M(I)) = 1
      if GRAPH.ON = 1
        show M(I) with "rmine.icn"
        let location.a(M(I)) = location.f(X,Y)
        display M(I) with "rmine.icn"
      endif
  loop
```

```
 63  endif
 64
 65  if I.MINES ne 0
 66    for I = P.MINES to (P.MINES + I.MINES)
 67       do
 68
 69       create MINE called M(I)
 70       MINE.NUM(M(I)) = I
 71       X = UNIFORM.F(-MF.WIDTH,MF.WIDTH,3)
 72       MINE.X(M(I)) = X
 73       Y = UNIFORM.F(MF.B,MF.T, 3)
 74       MINE.Y(M(I)) = Y
 75       RADIUS(M(I)) = I.RADIUS
 76       MINE.TYPE(M(I)) = 2
 77       MINE.STATUS(M(I)) = 1
 78         if GRAPH.ON = 1
 79           show M(I) with "bbmine.icn"
 80           let location.a(M(I)) = location.f(X,Y)
 81           display M(I) with "bbmine.icn"
 82         endif
 83    loop
 84  endif
 85
 86  if C.MINES ne 0
 87    for I = (P.MINES + I.MINES) to (P.MINES + I.MINES + C.MINES)
 88       do
 89
 90       create MINE called M(I)
 91       MINE.NUM(M(I)) = I
 92       X = UNIFORM.F(-MF.WIDTH,MF.WIDTH,3)
 93       MINE.X(M(I)) = X
 94       Y = UNIFORM.F(MF.B, MF.T, 3)
 95       MINE.Y(M(I)) = Y
 96       RADIUS(M(I)) = C.RADIUS
 97       MINE.TYPE(M(I)) = 3
 98       MINE.STATUS(M(I)) = 1
 99         if GRAPH.ON = 1
100           show M(I) with "blmine.icn"
101           let location.a(M(I)) = location.f(X,Y)
102           display M(I) with "blmine.icn"
103         endif
104    loop
105  endif
106
107  if T.MINES ne 0
108    for I = (P.MINES + I.MINES + C.MINES) to
109         (P.MINES + I.MINES + C.MINES + T.MINES)
110       do
111
112       create MINE called M(I)
113       MINE.NUM(M(I)) = I
114       X = UNIFORM.F(-MF.WIDTH,MF.WIDTH,3)
115       MINE.X(M(I)) = X
116       Y = UNIFORM.F(MF.B, MF.T, 3)
117       MINE.Y(M(I)) = Y
118       RADIUS(M(I)) = T.RADIUS
119       MINE.TYPE(M(I)) = 4
120       MINE.STATUS(M(I)) = 1
121         if GRAPH.ON = 1
122           show M(I) with "blmine.icn"
123           let location.a(M(I)) = location.f(X,Y)
124           display M(I) with "blmine.icn"
```

```
125         endif
126     loop
127   endif
128
129   if W.MINES ne 0
130     for I = (P.MINES + I.MINES + C.MINES + T.MINES) to
131           (P.MINES + I.MINES + C.MINES + T.MINES + W.MINES)
132       do
133
134       create MINE called M(I)
135       MINE.NUM(M(I)) = I
136       X = UNIFORM.F(-MF.WIDTH,MF.WIDTH,3)
137       MINE.X(M(I)) = X
138       Y = UNIFORM.F(MF.B, MF.T, 3)
139       MINE.Y(M(I)) = Y
140       RADIUS(M(I)) = W.RADIUS
141       MINE.TYPE(M(I)) = 5
142       MINE.STATUS(M(I)) = 1
143         if GRAPH.ON = 1
144           show M(I) with "blmine.icn"
145           let location.a(M(I)) = location.f(X,Y)
146           display M(I) with "blmine.icn"
147         endif
148     loop
149   endif
150
151   return
152
153   end ''MAKE.MINEFIELD
```

```
routine MAKE.ROUTE given I

''THIS ROUTINE IS DESIGNED TO GENERATE A ROUTE FOR A GIVEN ELEMENT.
''IF THE ELEMENT IS OF TYPE 1 (FULL PLOW), NO DEVIATION IS CALCULATED
''IN THE X DIMENSION OF THE CHECKPOINTS.


define I as an integer variable   ''identifies element
define J as an integer variable   ''identifies checkpoint
define K as an integer variable
define L as an integer variable
define DIVISOR as a real variable
define CUR.Y.LOC as a real variable


CUR.Y.LOC = 0.0              ''reset checkpoint y location counter

for L = 1 to NUM.CP          ''and then for each checkpoint
  do
                             ''generate, using a normal (0,1) distribution,
                             ''a navigation error in the x dimension
  select case ELEMENT.TYPE(E(I))

    case 1
      MOVEMENT.PLAN(I,L,1) = NORMAL.F(0.0, NAV.ERR.1, 3)
    case 2
      MOVEMENT.PLAN(I,L,1) = NORMAL.F(0.0, NAV.ERR.2, 3)
    case 3
      MOVEMENT.PLAN(I,L,1) = NORMAL.F(0.0, NAV.ERR.3, 3)
    case 4
      MOVEMENT.PLAN(I,L,1) = NORMAL.F(0.0, NAV.ERR.4, 3)
    case 5
      MOVEMENT.PLAN(I,L,1) = NORMAL.F(0.0, NAV.ERR.5, 3)
    default
      print 1 line thus
#####ERROR -- invalid type in initialize
  endselect

                                 ''first element has no navigation error
  if (I = 1)
    MOVEMENT.PLAN(I,L,1) = 0
  endif
                                 ''and assign a y location based on checkpoint
                                 ''sequence and interval
  MOVEMENT.PLAN(I,L,2) = CUR.Y.LOC
  CP(L) = CUR.Y.LOC
                                 ''and increment checkpoint y-location counter
  CUR.Y.LOC = CUR.Y.LOC + 20.0

loop

J = NUM.CP - 1

MOVEMENT.PLAN(I,1,5) = 1

for K = 1 to J                ''for each checkpoint, except the last

  do
                                ''calculate divisor of slope equation
  DIVISOR = MOVEMENT.PLAN(I,K+1,1) - MOVEMENT.PLAN(I,K,1)

                                ''if line not horizontal, (infinite slope)
```

```
                                ''calculate slope of line connecting checkpoint
                                ''with NEXT checkpoint and store result
                                ''in array
   if DIVISOR ne 0
     MOVEMENT.PLAN(I,K,3) =
       (MOVEMENT.PLAN(I,K+1,2) - MOVEMENT.PLAN(I,K,2)) / DIVISOR

                                ''otherwise, approximate infinite slope
    else
     MOVEMENT.PLAN(I,K,3) = 1000000
    endif

                                ''calculate y-intercept of line connecting
                                ''current checkpoint with NEXT checkpoint
                                ''and store result in array
    MOVEMENT.PLAN(I,K,4) =
           MOVEMENT.PLAN(I,K,2) - (MOVEMENT.PLAN(I,K,3) *
           MOVEMENT.PLAN(I,K,1))

loop

return

end ''MAKE.ROUTE
```

```
    1  routine MINE.DUMP
    2
    3  ''THIS ROUTINE DUMPS OUT ALL MINE DATA ASSOCIATED WITH A PARTICULAR ITERATION
    4  ''IT SHOULD NOT BE USED FOR MULTIPLE ITERATIONS AS THE OUTPUT FILE WILL
    5  ''BECOME EXTREMELY LARGE VERY FAST.
    6
    7
    8  print 3 lines thus

        DUMP OF MINE DATA


   12
   13  print 3 lines thus
             MINE DATA
  num      x        y    status  radius  type
  ----  -------  ------- ------  ------  ----
   17  for I = 1 to NUM.MINE
   18    print 1 line with I, MINE.X(M(I)), MINE.Y(M(I)), MINE.STATUS(M(I)),
   19      RADIUS(M(I)), MINE.TYPE(M(I)) thus
 ****  ***.***, ***.***   **     **.**   **
   21  start new output line
   22
   23  print 4 lines thus
           Mine vs Vehicle Pk Values
                 Mine Type
Veh Type\  1      2      3      4      5
-----------------------------------------
   28
   29  for J = 1 to 5
   30    do
   31    print 1 line with J, MINE.PK(1,J), MINE.PK(2,J), MINE.PK(3,J),
   32       MINE.PK(4,J), and MINE.PK(5,J) thus
 ***    *.**   *.**   *.**   *.**   *.**
   34  loop
   35
   36  return
   37
   38  end ''MINE.DUMP
```

```
 1  event MINE.ENCOUNTER given ELEMENT.ID, MINE.ID
 2
 3  ''THIS EVENT RESOLVES AN ENCOUNTER BETWEEN THE VEHICLE IDENTIFIED BY
 4  ''ELEMENT.ID AND THE MINE IDENTIFIED BY MINE.ID
 5
 6  define ELEMENT.ID as an integer variable    ''element in question
 7  define MINE.ID as an integer variable       ''mine in question
 8  define PK as a real variable                ''prob. of mine killing element
 9  define I as an integer variable             ''same as element.id
10  define J as an integer variable             ''same as mine.id
11  define K as an integer variable             ''element type
12  define L as an integer variable             ''mine type
13  define X,Y as real variables                ''location variables for graphics
14  define ROLL as a real variable              ''random draw
15  define ROLL2 as a real variable
16  define E.X and E.Y as real variables        ''element x and y (center front)
17
18  call UPDATE.LOCATION
19
20  I = ELEMENT.ID
21  J = MINE.ID
22  K = ELEMENT.TYPE(E(I))
23  L = MINE.TYPE(M(J))
24
25  SCH.EVENTS(I) = 0
26  EVENT.LIST(I) = 0
27
28  PK = 0
29                                  ''if mine is active, lookup appropriate Pk
30  if MINE.STATUS(M(J)) ne 0
31    PK = MINE.PK(L,K)
32  endif
33                                  ''random number determination
34  ROLL = uniform.f(0.0,1.0,5)
35  ROLL2 = uniform.f(0.0,1.0,6)
36                                  ''if full plow versus non-contact mine
37                                  ''determine if plow catches mine, if yes
38                                  ''call full.plow to move mine
39  if (ELEMENT.TYPE(E(I)) = 1) and (MINE.TYPE(M(J)) ne 3)
40    if ROLL2 < PLOW.EFFECT(E(I))
41      call FULL.PLOW giving I,J
42                                  ''increment global MINE.MOVED counter
43      MINE.MOVED = MINE.MOVED + 1
44                                  ''mine moved, so skip remainder of routine
45      go to 'PLOWED'
46    endif
47  endif
48                                  ''determine if mine destroys element, if so
49                                  ''update mine counters, element status,
50                                  ''element speed and record time of death
51  if ROLL le PK
52
53    MINE.KILLS = MINE.KILLS + 1.0
54    TOTAL.LOST = TOTAL.LOST + 1.0
55    TIME.OF.DEATH(I) = time.v
56
57    if STATUS(E(I)) = 2
58       LOST.TO.MINES.BYPASSING = LOST.TO.MINES.BYPASSING + 1
59    endif
60
61    select case MINE.TYPE(M(J))
62      case 1  ''pressure mine
```

```
   63        MINE.KILL.1 = MINE.KILL.1 + 1.0
   64      case 2  ''influence mine
   65        MINE.KILL.2 = MINE.KILL.2 + 1.0
   66      case 3  ''contact mine
   67        MINE.KILL.3 = MINE.KILL.3 + 1.0
   68      case 4  ''mine type 4
   69        MINE.KILL.4 = MINE.KILL.4 + 1.0
   70      case 5  ''mine type 5
   71        MINE.KILL.5 = MINE.KILL.5 + 1.0
   72      default
   73        print 1 line thus
      #####ERROR -- invalid mine type encountered
   75    endselect
   76
   77                                  ''if contest was working plow vs contact mine,
   78                                  ''then kill records are in error, to correct
   79                                  ''reduce plow effectiveness, correct mine
   80                                  ''counter, causualty counter, time of death
   81                                  ''record and skip remainder of routine
   82    if (ELEMENT.TYPE(E(I)) = 1) and (MINE.TYPE(M(J)) = 3) and
   83          (P.STATUS(E(I)) ne 0)
   84      PLOW.EFFECT(E(I)) = 0
   85      P.STATUS(E(I)) = 0
   86      MINE.STATUS(M(J)) = 0
   87      MINE.KILL.3 = MINE.KILL.3 - 1.0
   88      MINE.KILLS = MINE.KILLS - 1.0
   89      TOTAL.LOST = TOTAL.LOST - 1.0
   90      if STATUS(E(I)) = 2
   91        LOST.TO.MINE.BYPASSING = LOST.TO.MINE.BYPASSING - 1
   92      endif
   93
   94      TIME.OF.DEATH(I) = 0
   95
   96      if GRAPH.ON = 1
   97        erase M(J)
   98        show M(J) with "blgmine.icn"
   99        x = MINE.X(M(J))
  100        y = MINE.Y(M(J))
  101        let location.a(M(J)) = location.f(X,Y)
  102      endif
  103      go to 'PLOWED'
  104    endif
  105
  106    X = ELEMENT.X(E(I))
  107    Y = ELEMENT.Y(E(I))
  108    E.X = X
  109    E.Y = Y
  110
  111    if GRAPH.ON = 1
  112      show OB(I) with "obs1.icn"
  113      if ELEMENT.TYPE(E(I)) = 1
  114        show E(I) with "dfwplow.icn"
  115       else
  116        show E(I) with "tank.icn"
  117      endif
  118      let location.a(OB(I)) = location.f(X,Y)
  119      let location.a(E(I)) = location.f(X,Y)
  120    endif
  121                                  ''update mine status
  122    MINE.STATUS(M(J)) = 0
  123    if GRAPH.ON = 1
  124      erase M(J)
```

```
  125      show M(J) with "gmine.icn"
  126      x = MINE.X(M(J))
  127      y = MINE.Y(M(J))
  128      let location.a(M(J)) = location.f(X,Y)
  129    endif
  130
  131    STATUS(E(I)) = 0
  132    SPEED(E(I)) = 0
  133    OBSTACLE.X(OB(I)) = E.X ''X1
  134    OBSTACLE.Y(OB(I)) = E.Y ''Y1
  135    OBS.RADIUS(OB(I)) = ELEM.RADIUS(E(I))
  136
  137    call OBSTACLE.CONSOLIDATION
  138    call CALENDAR.UPDATE
  139
  140  else
  141
  142    MINE.STATUS(M(J)) = 0
  143    if GRAPH.ON = 1
  144      erase M(J)
  145      show M(J) with "gmine.icn"
  146      x = MINE.X(M(J))
  147      y = MINE.Y(M(J))
  148      let location.a(M(J)) = location.f(X,Y)
  149    endif
  150
  151  'PLOWED'
  152    call NEXT.ENCOUNTER giving I
  153  endif
  154
  155  return
  156
  157  end ''MINE.ENCOUNTER  edited 31 August 1991
```

```
 1  event NEW.CP given ID
 2
 3  ''THE EVENT CHANGES THE MOVEMENT PATH USED BY THE MOVING ELEMENT BASED
 4  ''UPON ITS PROGRESS IAW THE PREPARED MOVEMENT PLAN.
 5
 6  define ID as an integer variable
 7  define CP.ID as an integer variable
 8
 9  call UPDATE.LOCATION
10
11  SCH.EVENTS(ID) = 0
12  EVENT.LIST(ID) = 0
13
14  call FIND.CURRENT.CP giving ID yielding CP.ID
15
16                              ''element is in normal mode
17  if STATUS(E(ID)) = 1
18    if CP.ID + 1  = NUM.CP    ''all done moving
19       if SPEED(E(ID)) ne 0,  ''not already recorded (this could be
20                              ''a calendar update....)
21         if (TIME.OF.COMPLETION(ID) = 0) and (TIME.OF.DEATH(ID) = 0)
22           TIME.OF.COMPLETION(ID) = time.v
23         endif
24       endif
25       SPEED(E(ID)) = 0
26       return
27    endif
28
29                              ''update checkpoint status flags
30    MOVEMENT.PLAN(ID,CP.ID,5) = 0
31    MOVEMENT.PLAN(ID,CP.ID + 1,5) = 1
32
33                              ''update element position location
34    ELEMENT.X(E(ID)) = MOVEMENT.PLAN(ID,CP.ID + 1,1)
35    ELEMENT.Y(E(ID)) = MOVEMENT.PLAN(ID,CP.ID + 1,2)
36
37  endif
38                              ''if element in bypass mode
39  if STATUS(E(ID)) = 2,
40    if CP.ID = 3              ''and ready to re-enter movement plan from bypass
41      ELEMENT.X(E(ID)) = BYPASS.MAP(ID,4,1)
42      ELEMENT.Y(E(ID)) = BYPASS.MAP(ID,4,2)
43      STATUS(E(ID)) = 1
44
45                              ''case where bypass include exit boundary
46      if ELEMENT.Y(E(ID)) = CP(NUM.CP)
47        SPEED(E(ID)) = 0.0
48        if (TIME.OF.COMPLETION(ID) = 0) and (TIME.OF.DEATH(ID) = 0)
49          TIME.OF.COMPLETION(ID) = time.v
50        endif
51      endif
52
53      for I = 1 to NUM.CP     ''erase checkpoint status indicator for element
54        do
55        MOVEMENT.PLAN(ID,I,5) = 0
56      loop
57
58                              ''determine which checkpoint was re-entered into
59                              ''and mark that one as the current checkpoint
60      for I = 1 to NUM.CP-1
61        do
62        if ((CP(I) < ELEMENT.Y(E(ID))) and (CP(I+1) > ELEMENT.Y(E(ID))))
```

```
   63          MOVEMENT.PLAN(ID,I,5) = 1
   64         else
   65          MOVEMENT.PLAN(ID,NUM.CP,5) = 1
   66        endif
   67      loop
   68
   69      call NEXT.ENCOUNTER giving ID
   70      return
   71    endif
   72
   73    ELEMENT.X(E(ID)) = BYPASS.MAP(ID,CP.ID+1,1)
   74    ELEMENT.Y(E(ID)) = BYPASS.MAP(ID,CP.ID+1,2)
   75    BYPASS.MAP(ID,CP.ID, 5) = 0
   76    BYPASS.MAP(ID,CP.ID+1, 5) = 1
   77  endif
   78
   79  call NEXT.ENCOUNTER giving ID
   80
   81  return
   82
   83  end ''NEW.CP edited 5 Sept 91
```

```
routine NEXT.ENCOUNTER given ID

''THIS ROUTINE DETERMINES THE NEXT ENCOUNTER EVENT FOR ELEMENT ID.

define ID as an integer variable
define MINE.ID as an integer variable
define CP.DISTANCE as a real variable
define M.DISTANCE as a real variable
define OBS.DISTANCE as a real variable
define DURATION as a double variable
define SCH.TIME as a double variable

if (STATUS(E(ID)) ne 0) and (SPEED(E(ID)) ne 0)


''### ERROR TRAP -- simulation currently should not run longer than
''10 simulated minutes maximum. This statement will stop the model if
''simulation time greatly exceed this value.

  if (time.v > 25.00)

    print 1 line thus
run ran for 25.00 min. and was stopped

    schedule a STOP.SIM now
  endif

                              ''determine next manuver event
                              ''for each type of event
  call DISTANCE.TO.CP giving ID yielding CP.DISTANCE, X1, X2
  call DISTANCE.TO.MINE giving ID yielding MINE.ID, M.DISTANCE
  call DISTANCE.TO.OBS giving ID yielding OBS.DISTANCE


                              ''schedule the closest of the
                              ''possible manuver events
  if CP.DISTANCE < min.f(M.DISTANCE, OBS.DISTANCE)

                               ''schedule a cp encounter
    call DELTA.TIME giving ID, CP.DISTANCE yielding DURATION
    SCH.TIME = time.v + DURATION
    schedule a NEW.CP called SCH.EVENTS(ID) giving ID at SCH.TIME
    EVENT.LIST(ID) = 1          ''update event log. Event log is
                                ''array to track next event of each
                                ''element. 1 = CP, 2 = Mine, 3 = OBS

  endif


  if M.DISTANCE < min.f(CP.DISTANCE, OBS.DISTANCE)
                               ''schedule mine encounter
    call DELTA.TIME giving ID, M.DISTANCE yielding DURATION
    SCH.TIME = time.v + DURATION
    schedule a MINE.ENCOUNTER called SCH.EVENTS(ID) giving ID,
      MINE.ID at SCH.TIME
    EVENT.LIST(ID) = 2

  endif

  if OBS.DISTANCE < min.f(M.DISTANCE, CP.DISTANCE)
                               ''schedule obstacle encounter
    call DELTA.TIME giving ID, OBS.DISTANCE yielding DURATION
```

```
   63      SCH.TIME = time.v + DURATION
   64      schedule an OBSTACLE.ENCOUNTER called SCH.EVENTS(ID)
   65         giving ID at SCH.TIME
   66      EVENT.LIST(ID) = 3
   67
   68    endif
   69  endif
   70
   71  return
   72
   73  end ''NEXT.ENCOUNTER
```

```
routine OBSTACLE.CONSOLIDATION

''THIS ROUTINE IS DESIGNED TO CONSOLIDATE OBSTACLES WHICH ARE TOO CLOSE
''TOGETHER TO REASONABLY PERMIT MOVEMENT BETWEEN THEM (IN THE CASE OF
''VEHICLES DISABLED WHILE BYPASSING) OR ARE SO CLOSE TOGETHER ON THE LANE
''THAT ATTEMPTING TO REGAIN THE LANE WOULD BE SENSELESS. THE ROUTINE WORKS
''BY CONDUCTING A PAIRWISE COMPARISON OF ALL OBSTACLES, COMBINING THOSE
''WHICH MEET THE ABOVE MENTIONED CONDITIONS. IF A PAIR OF OBSTACLES IS
''COMBINED, THEN THE ROUTINE RESTARTS FROM THE BEGINNING. ONCE A PAIR OF
''OBSTACLES HAS BEEN COMBINED, THE RESULTING OBSTACLE IS LABELED WITH THE
''LOWEST NUMBER BETWEEN THE TWO AND THE 2ND NUMBERS DATA IS ZERO'ED OUT.

define I,J as integer variables
define DISTANCE as a real variable
define RAD1 as a real variable
define X1, Y1 as real variables
define RAD2 as a real variable
define X2, Y2 as real variables
define RAD as a real variable
define X, Y as real variables
define X.EDGE and Y.EDGE as real variables

'RESTART'
for I = 1 to NUM.OBSTACLE
  do
                                ''if obstacle exists, record needed info
  if (OBSTACLE.Y(OB(I)) > 0) and (OBS.RADIUS(OB(I)) > 0)
    RAD1 = OBS.RADIUS(OB(I))
    X1 = OBSTACLE.X(OB(I))
    Y1 = OBSTACLE.Y(OB(I))
                                ''compare to all remaining obstacles
    for J = (I+1) to NUM.OBSTACLE
      do
      if (OBSTACLE.Y(OB(I)) > 0) and (OBS.RADIUS(OB(J)) > 0)
        RAD2 = OBS.RADIUS(OB(J))
        X2 = OBSTACLE.X(OB(J))
        Y2 = OBSTACLE.Y(OB(J))
                                ''compute distance between obstacle centers
        DISTANCE = sqrt.f((X1-X2)**2 + (Y1-Y2)**2)

        if DISTANCE < 2*(RAD1 + RAD2),

          RAD = (DISTANCE + RAD1 + RAD2)/2.0

          if (X1 < X2)
            X.EDGE = X1 - RAD1
            X = X.EDGE + RAD
          endif

          if (X2 <= X1)
            X.EDGE = X2 - RAD2
            X = X.EDGE + RAD
          endif

          if (Y1 < Y2)
            Y.EDGE = Y1 - RAD1
            Y = Y.EDGE + RAD
          endif

          if (Y2 <= Y1)
            Y.EDGE = Y2 - RAD2
            Y = Y.EDGE + RAD
```

```
   63            endif
   64
   65            OBSTACLE.X(OB(I)) = X
   66            OBSTACLE.Y(OB(I)) = Y
   67            OBS.RADIUS(OB(I)) = RAD
   68
   69                                  ''eliminate 2nd obstacle from obstacle list
   70            OBSTACLE.X(OB(J)) = 0.0
   71            OBSTACLE.Y(OB(J)) = 0.0
   72            OBS.RADIUS(OB(J)) = 0.0
   73            go to 'RESTART'
   74          endif
   75        endif
   76      loop
   77    endif
   78  loop
   79
   80  return
   81
   82  end ''OBSTACLE.CONSOLIDATION
```

```
 1  event OBSTACLE.ENCOUNTER given ID
 2
 3  ''THIS EVENT CAUSES THE STATUS OF THE ELEMENT TO CHANGE TO BYPASS AND
 4  ''TO BEGIN USING THE BYPASS MAP FOR MOVEMENT
 5
 6  define ID as an integer variable
 7
 8  call UPDATE.LOCATION
 9
10  SCH.EVENTS(ID) = 0
11  EVENT.LIST(ID) = 0
12  STATUS(E(ID)) = 2              ''change element status to 'bypass'
13
14  BYPASS.MAP(ID, 1, 5) = 1    ''point to 1st checkpoint in bypass
15  call NEXT.ENCOUNTER giving ID
16
17  return
18
19  end ''OBSTACLE.ENCOUNTER
```

```
    1  routine OUTPUT
    2
    3  ''THIS ROUTINE CAUSES THE OUTPUT OF THE INPUT VALUES AND THE SUMMARY DATA
    4  ''GENERATED DURING A MODEL RUN
    5
    6  print 3 lines thus

        OUTPUT DATA

   10
   11  print 2 lines with MINE.MOVED thus

FULL WIDTH PLOWS moved ***.** mines.
   14
   15  print 2 lines thus

MINE EFFECTS
   18
   19  print 6 lines with MINE.KILL.1, MINE.KILL.2, MINE.KILL.3, MINE.KILL.4,
   20  MINE.KILL.5, MINE.KILLS thus
Type 1 mines had ***.** kills (pressure mines)
Type 2 mines had ***.** kills (influence mines)
Type 3 mines had ***.** kills (contact mines)
Type 4 mines had ***.** kills
Type 5 mines had ***.** kills
Total mine kills were ****.**
   27
   28  print 2 lines thus

Red Overwatching Direct Fire Data
   31  print 3 lines with R.O.SHOTS, R.O.KILL thus

Red fired ****.* direct fire shots.
accomplishing ***.** kills of blue taskforce vehicles.
   35
   36  return
   37  end ''OUTPUT
```

```
routine R.DIRECT.OVERWATCH given RO.ID

''THIS ROUTINE DETERMINES WHEN A RED DIRECT FIRE EVENT SHOULD TAKE PLACE

define RO.ID as an integer variable
define RATE as a real variable
define ACQUIRE as a real variable
define SHOT.TIME as a double variable
define X, Y as real variables

X = R.O.X(RO(RO.ID))
Y = R.O.Y(RO(RO.ID))

if GRAPH.ON = 1
  show RO(RO.ID) with "rotank.icn"
  let location.a(RO(RO.ID)) = location.f(X,Y)
endif

if R.O.STATUS(RO(RO.ID)) = 1
  RATE = R.O.BETA(RO(RO.ID))
  ACQUIRE = exponential.f(.5, 7)
  SHOT.TIME = RATE + ACQUIRE
endif

schedule a DIRECT.FIRE giving RO.ID at time.v + SHOT.TIME

return

end ''end R.DIRECT.OVERWATCH
```

```
routine SET.DISPLAY

''THIS ROUTINE IS USED TO INITIALIZE THE "WORLD" THAT IS USED BY THE
''GRAPHICS ROUTINES IN SIMGRAPHICS

define DISPLAY.DEPTH as a real variable
define START.PT as a real variable

DISPLAY.DEPTH = NUM.CP * CP.INTERVAL + 10.0
START.PT = -10.0

let timescale.v = 100
let vxform.v = 1
call setworld.r( -(.5 * DISPLAY.DEPTH), (.5 * DISPLAY.DEPTH), START.PT,
DISPLAY.DEPTH)

return
end ''SET.DISPLAY
```

```
 1  event START given ID
 2
 3  ''THIS EVENT INITIALIZES THE MOVEMENT OF INDIVIDUAL TASKFORCE ELEMENTS
 4
 5  define ID as an integer variable
 6  define CONVERSION.FACTOR as a real variable
 7
 8  CONVERSION.FACTOR = 1000.0/60.0
 9  SPEED(E(ID)) = E.SPEED * CONVERSION.FACTOR
10
11  if ID = 1
12    SPEED(E(ID)) = E.SPEED.1 * CONVERSION.FACTOR
13  endif
14
15  call NEXT.ENCOUNTER giving ID
16
17  return
18  end ''START
```

```
    1  event STOP.SIM
    2
    3  ''USED TO STOP THE SIMULATION UNDER USER DEFINED CONDITIONS
    4  ''DUMPS ALL DATA TO OUTPUT FILE
    5
    6  print 1 line thus
SIMULATION halted due to time expiration
    8
    9  call DATA.DUMP
   10  call MINE.DUMP
   11  call OUTPUT
   12
   13  STOP
   14
   15  end ''event STOP.SIM
```

```
    1  routine SUMMARY
    2
    3  ''DATA INPUT VALUES AND MODEL OUTPUTS
    4
    5  define J as an integer variable
    6
    7  print 3 lines thus

        SUMMARY DATA

   11
   12  print 9 lines with NUM.ELEMENT, F.PLOWS, T.PLOWS, TRACK1, TRACK2, TRACK3 thus
     TASKFORCE SIZE
     --------- ----
Total number of elements was ****
Number of Full Plows = ***
Number of Track Plows = ***
Number of Track Type 1 = ***
Number of Track Type 2 = ***
Number of Track Type 3 = ***

   22
   23
   24  print 3 lines with E.SPEED.1, E.SPEED thus
     Taskforce Velocity
Full plow has a velocity of ***.*** kph
Track type 1 has a velocity of ***.*** kph
   28
   29  print 8 lines with P.MINES, I.MINES, C.MINES, T.MINES, and W.MINES thus
       MINE DATA

Number of Pressure Mines = ****
Number of Influence Mines = ****
Number of Contact Mines = ****
Number of Type 4 Mines = ****
Number of Type 5 Mines = ****

   38
   39  print 18 lines with AREA.DENSITY.TYPE.1, AREA.DENSITY.TYPE.2,
   40  AREA.DENSITY.TYPE.3, AREA.DENSITY.TYPE.4, AREA.DENSITY.TYPE.5, AREA.DENSITY,
   41  LINEAR.DENSITY.TYPE.1, LINEAR.DENSITY.TYPE.2, LINEAR.DENSITY.TYPE.3,
   42  LINEAR.DENSITY.TYPE.4, LINEAR.DENSITY.TYPE.5, LINEAR.DENSITY thus

            MINE DENSITIES
AREA DENSITY
   TYPE 1 MINES = ***.***
   TYPE 2 MINES = ***.***
   TYPE 3 MINES = ***.***
   TYPE 4 MINES = ***.***
   TYPE 5 MINES = ***.***
TOTAL ALL MINES = ***.***

LINEAR DENSITY
   TYPE 1 MINES = ***.***
   TYPE 2 MINES = ***.***
   TYPE 3 MINES = ***.***
   TYPE 4 MINES = ***.***
   TYPE 5 MINES = ***.***
TOTAL ALL MINES = ***.***

   61
   62  print 4 lines thus
```

```
           Mine vs Vehicle Pk Values
                 Mine Type
Veh Type\  1      2      3      4      5
-----------------------------------------
```

```
   67
   68  for J = 1 to 5
   69    do
   70    print 1 line with J, MINE.PK(1,J), MINE.PK(2,J), MINE.PK(3,J),
   71       MINE.PK(4,J), and MINE.PK(5,J) thus
  ***    *.**   *.**   *.**   *.**   *.**
   73  loop
   74
   75  print 3 lines thus
RED OVERWATCH
TYPE     STATUS    FIRE RATE   MEAN ACQ RATE
----     ------    ---------   -------------
   79  for J = 1 to NUM.RED.OVERWATCH
   80    do
   81  print 1 line with R.O.TYPE(RO(J)), R.O.STATUS(RO(J)), R.O.RATE(RO(J)),
   82    R.O.BETA(RO(J)) thus
 **       **       ***.***       ***.***
   84  loop
   85
   86  print 1 line with NUMBER.OF.RUNS thus
The simulation made *** runs.
   88
   89
   90  print 1 line with PLOW.WIDTH.1, PLOW.EFFECT.1 thus
The full width plow used had a width of ***.** and an effectiveness of ***.**
   92
   93
   94
   95  print 3 lines thus

            SUMMARY STATISTICS

   99
  100  print 1 line thus
           MINES MOVED BY PLOW
  102
  103  print 2 lines with MEAN.MINE.MOVED thus
An average of ***.** mines were displaced by the plow each run

  106
  107  print 1 line thus
           MINE KILLS BY MINE TYPE
  109  print 8 lines with MEAN.MINE.KILL.1, VAR.MINE.KILL.1, MEAN.MINE.KILL.2,
  110  VAR.MINE.KILL.2, MEAN.MINE.KILL.3, VAR.MINE.KILL.3, MEAN.MINE.KILL.4,
  111  VAR.MINE.KILL.4, MEAN.MINE.KILL.5, VAR.MINE.KILL.5, MEAN.MINE.KILLS,
  112  VAR.MINE.KILLS thus

Mean mine kills by type 1 mines was ***.**, with variance ****.**
Mean mine kills by type 2 mines was ***.**, with variance ****.**
Mean mine kills by type 3 mines was ***.**, with variance ****.**
Mean mine kills by type 4 mines was ***.**, with variance ****.**
Mean mine kills by type 5 mines was ***.**, with variance ****.**
Mean mine kills for all mine types ****.**, with variance ****.**

  121
  122  print 1 line thus
           DIRECT FIRE STATS
  124
```

```
  125  print 7 lines with MEAN.R.O.SHOTS, VAR.R.O.SHOTS, MEAN.R.O.KILL,
  126  VAR.R.O.KILL thus

Red direct fire overwatch fired a mean of ****.** with a variance of
  ****.** shots per run.

Red direct fire overwatch killed a mean of ***.** with a variance of
  ****.** taskforce vehicles per run.

  134
  135  print 3 lines with MEAN.LOSS.RATE, VAR.LOSS.RATE thus

The mean taskforce loss rate was ****.** with a variance of ****.**

  139  return
  140  end ''SUMMARY
```

```
routine UPDATE.LOCATION

''THIS ROUTINE CAUSES THE IDENTIFIED ELEMENT TO BE MOVED ALONG ITS
''PREDISIGNATED MOVEMENT PATH A DISTANCE DETERMINED AS A FUNCTION
''OF TIME PASSAGE SINCE THE LAST UPDATE AND THE ELEMENT VELOCITY.
''THE ROUTINE WORKS BY FIRST DETERMINING WHAT PATH TO USE, THEN
''CALCULATING THE COORDINATES OF THE MOVE RESULT, THEN UPDATING
''THE POSITION LOCATION FIELDS OF THE ELEMENT

define ID as an integer variable                ''element index
define DISTANCE as a real variable              ''distance to next cp
define DISTANCE.TO.TRAVEL as a real variable    ''travel distance
define TIME.PASSAGE as a double variable
define X as a real variable
define Y as a real variable
define X1 as a real variable                     ''delta x to next cp
define Y1 as a real variable                     ''delta y to next cp
define X2 as a real variable                     ''movement in x
define Y2 as a real variable                     ''movement in y
define MOV.FRACTION as a real variable          ''element movement as
                                                ''a fraction of distance
                                                ''to next checkpoint

TIME.PASSAGE = time.v - OLD.TIME                ''time since last update
OLD.TIME = time.v

for ID = 1 to NUM.ELEMENT
  do
                                  ''make sure in bounds, if not, stop it.
  if ELEMENT.Y(E(ID)) >= CP(NUM.CP)
    SPEED(E(ID)) = 0
                                  ''update time to traverse record
    if STATUS(E(ID)) = 1 or STATUS(E(ID)) = 2
      if (TIME.OF.COMPLETION(ID) = 0) and (TIME.OF.DEATH(ID) = 0)
        TIME.OF.COMPLETION(ID) = time.v
      endif
    endif
  endif


  if ((STATUS(E(ID)) = 1) or (STATUS(E(ID)) = 2))  ''living vehicles
   if SPEED(E(ID)) > 0                             ''that are moving
                                                   ''determine how far?
    call DELTA.DISTANCE giving ID and TIME.PASSAGE yielding
        DISTANCE.TO.TRAVEL



                              ''determine distance to next checkpoint
                              ''and the delta x, delta y
    call DISTANCE.TO.CP giving ID yielding DISTANCE, X1, Y1

                              ''case where element distance from
                              ''cp so small interpreted as 0, make
                              ''a very small, but valid distance to
                              ''prevent division by 0
    if (DISTANCE = 0)
       DISTANCE = .00001
    endif
                              ''compute the fraction of the distance
                              ''to the next checkpoint that the required
                              ''move will cover
```

```
   63      MOV.FRACTION = DISTANCE.TO.TRAVEL / DISTANCE
   64
   65                                ''translate that fraction into x and y
   66                                ''movement
   67      X2 = X1 * MOV.FRACTION
   68      Y2 = Y1 * MOV.FRACTION
   69                                ''and add that movement to the current
   70                                ''element position
   71      X = ELEMENT.X(E(ID)) + X2
   72      Y = ELEMENT.Y(E(ID)) + Y2
   73
   74      ELEMENT.X(E(ID)) = X
   75      ELEMENT.Y(E(ID)) = Y
   76
   77                                ''graphics on/of flag
   78      if GRAPH.ON = 1
   79        if ELEMENT.TYPE(E(ID)) = 1
   80          if P.STATUS(E(ID)) = 1
   81            show E(ID) with "fwplow.icn"
   82          endif
   83
   84          if P.STATUS(E(ID)) = 0
   85            show E(ID) with "xfwplow.icn"
   86          endif
   87
   88          let location.a(E(ID)) = location.f(X,Y)
   89        endif
   90
   91        if ELEMENT.TYPE(E(ID)) = 3
   92          show E(ID) with "tank1.icn"
   93          let location.a(E(ID)) = location.f(X,Y)
   94        endif
   95
   96        if ELEMENT.TYPE(E(ID)) = 4
   97          show E(ID) with "tank2.icn"
   98          let location.a(E(ID)) = location.f(X,Y)
   99        endif
  100      endif
  101
  102    endif
  103   endif
  104  loop
  105
  106  return
  107  end ''UPDATE.LOCATION edited 5 Sept 91
```

APPENDIX D - GRAPHIC ICONS

This appendix contains diagrams depicting the icons used in the graphics portion of the model. The icons are included because although the SIMSCRIPT code will run on a variety of computer systems, the SIMGRAPHICS portion of the code is unique to the type of computer system it was generated on. To allow interested users the ability to recreate the graphics on their specific platforms, the icons are reproduced here. The text strings ending in "icn" at the base of each diagram are the file names which the program looks for when using that particular icon. In the cases where multiple text strings appear beneath the icon (Figure D-2), multiple icons of this shape are used, each of a different color. Any of the icons may be of any color the user chooses. It is suggested that the bottom row of mine icons depicted in Figure D-6 be of the same color as these icons are used to depict inactive mines.

The "crosshairs" on each icon diagram identify the center of the icon which must be identified to the SIMGRAPHICS editor when constructing the icon. The measurement given at the bottom of each diagram provides the "scale" of the icon which must also be input to the editor before the icon is used.

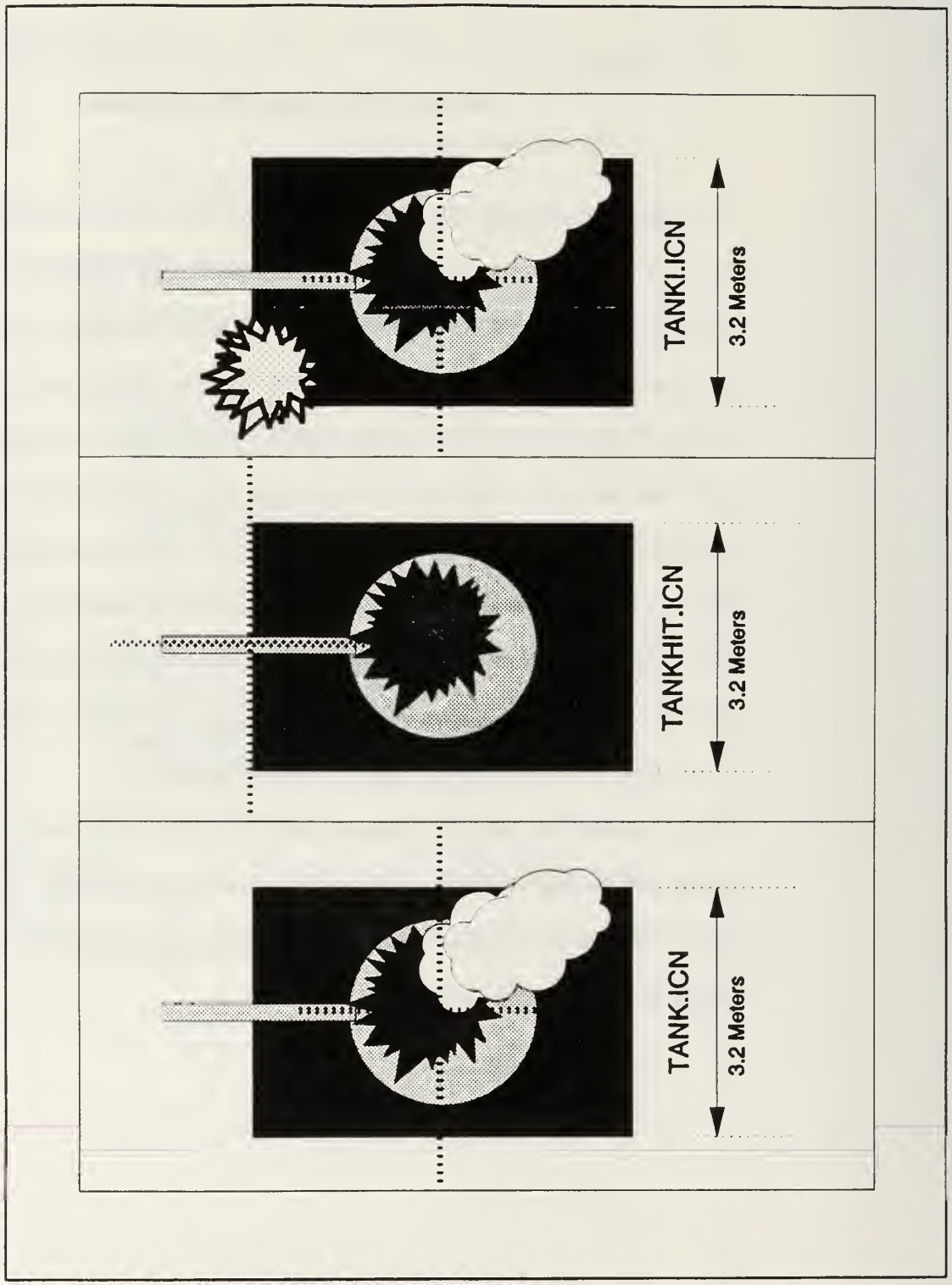

Figure D-1 MINEFIELD MODEL ICONS

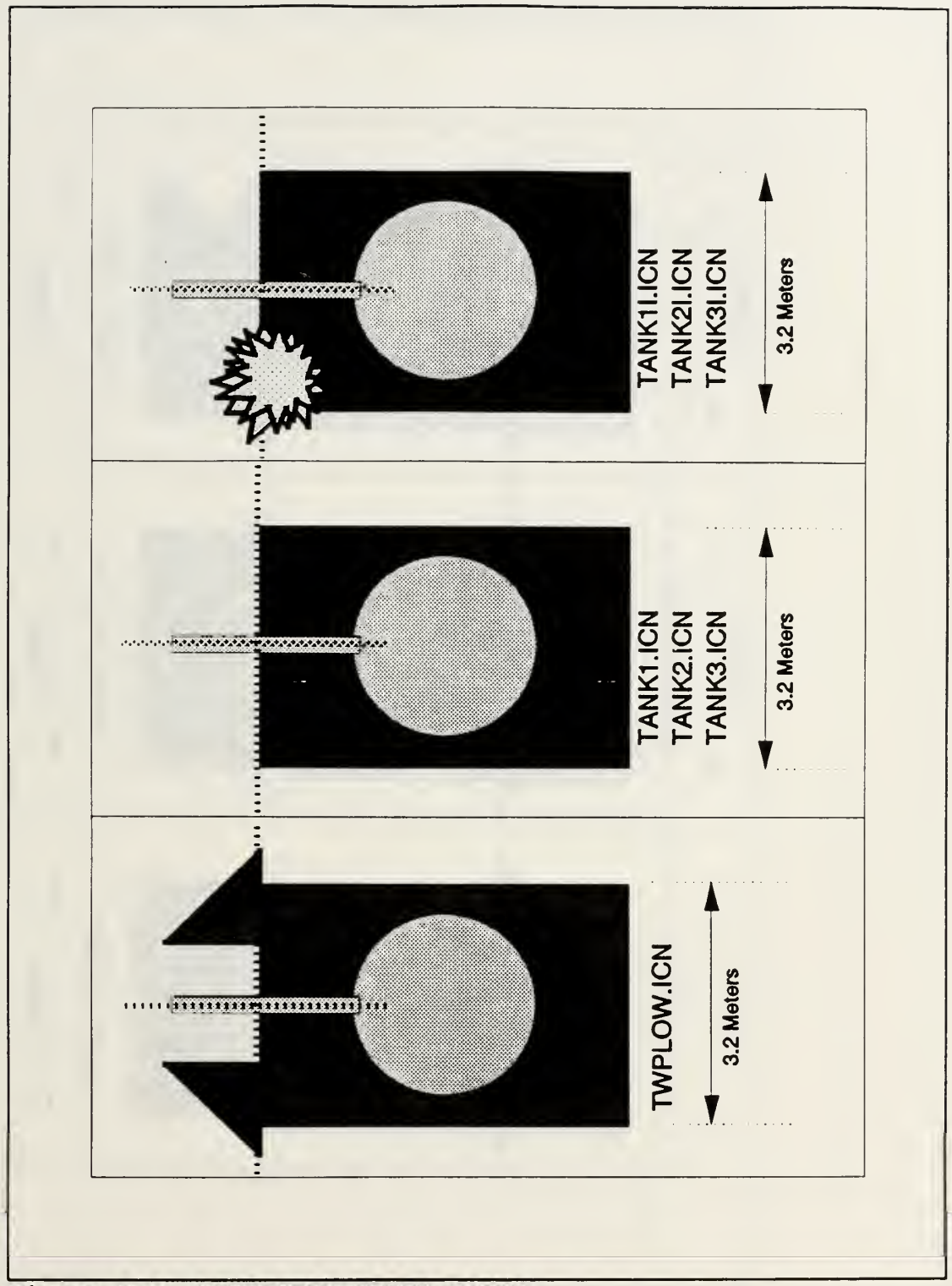

Figure D-2 MINEFIELD MODEL ICONS

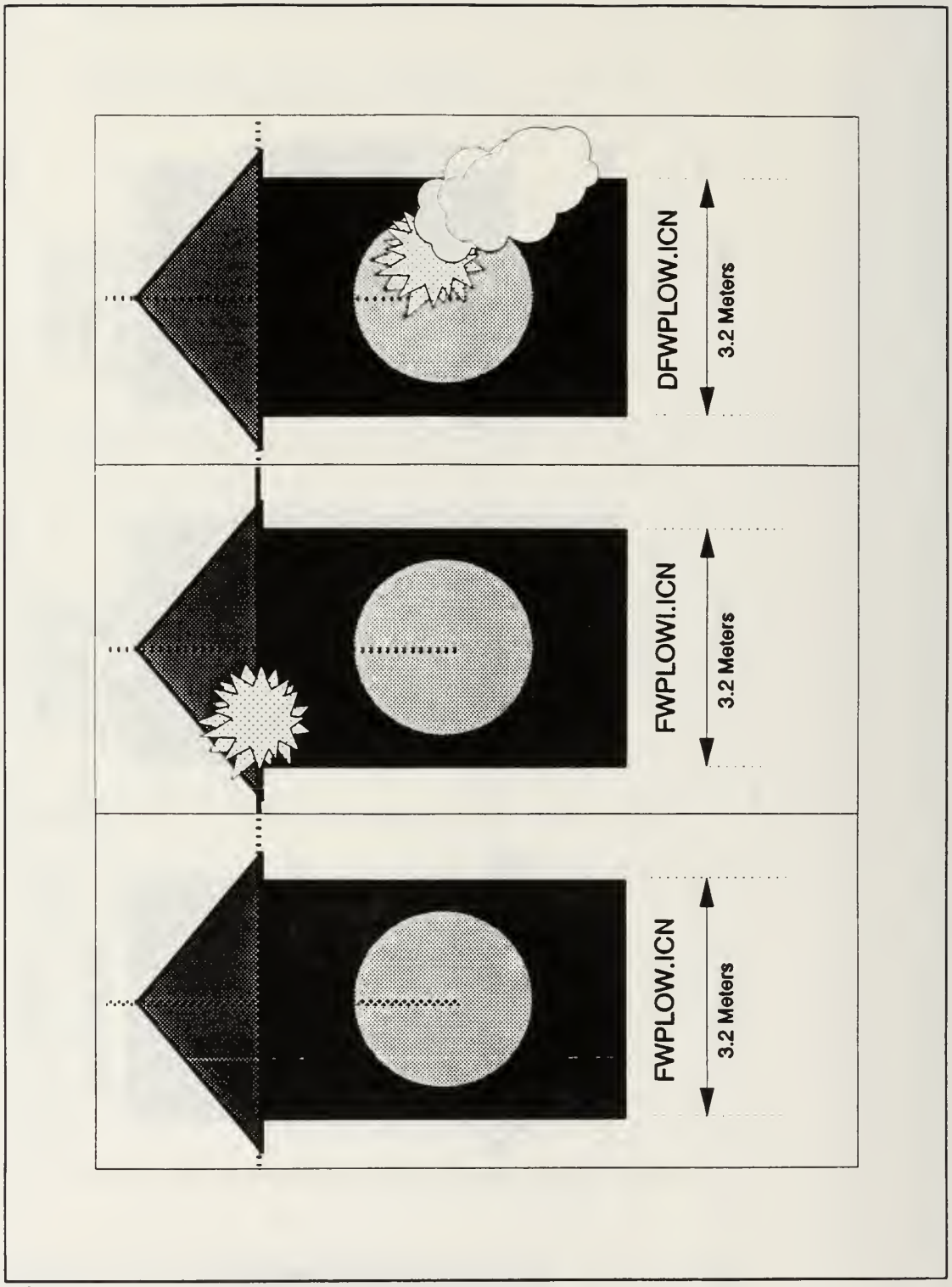

Figure D-3 **MINEFIELD MODEL ICONS**

Figure D-4 **MINEFIELD MODEL ICONS**

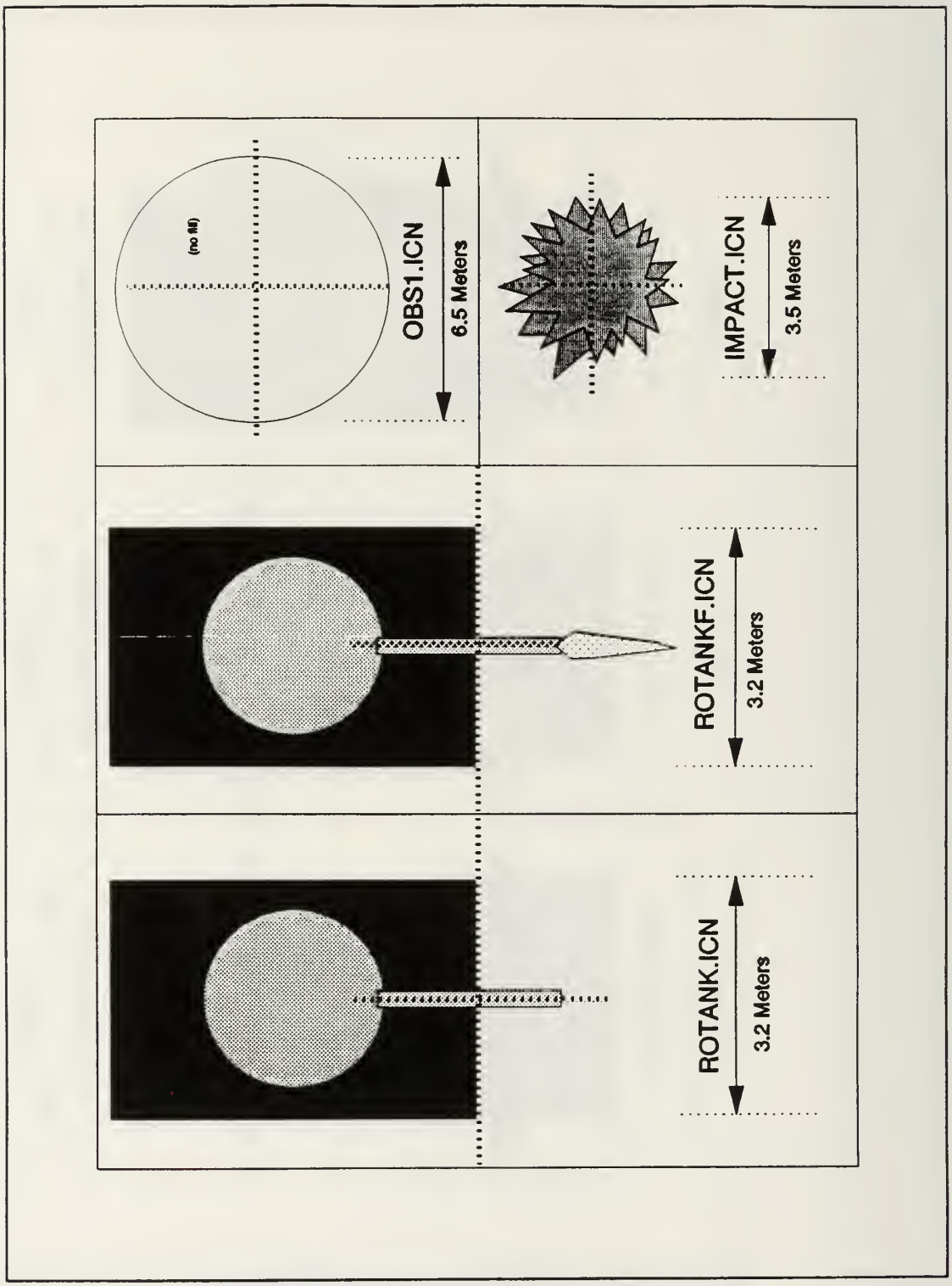

Figure D-5 MINEFIELD MODEL ICONS

Figure D-6 **MINEFIELD MODEL ICONS**

APPENDIX E - EXAMPLE SINGLE AVAILABLE OUTPUTS

A large amount of data can be extracted from each model iteration if desired. The following is an example of the data that may be recorded by the user for each iteration.

DUMP OF MINEFIELD DATA

MINE DATA

num	x	y	status	radius	type
1	-56.762,	82.133	1	.10	1
2	67.510,	79.356	1	.10	1
3	-37.463,	88.430	1	.10	1
4	56.123,	100.501	1	.10	1
5	16.748,	89.538	1	.10	1
6	5.149,	93.052	1	.10	1
7	17.030,	92.072	1	.10	1
8	44.720,	89.041	1	.10	1
9	5.005,	87.634	1	.10	1
10	-16.053,	99.971	1	.10	1
11	37.485,	80.661	1	.10	1
12	-70.096,	99.896	1	.10	1
13	63.482,	84.333	1	.10	1
14	28.201,	95.031	1	.10	1
15	46.276,	84.887	1	.10	1
16	-64.704,	96.522	1	.10	1
17	20.460,	97.545	1	.10	1
18	56.951,	88.502	1	.10	1
19	-61.047,	80.087	1	.10	1
20	-4.905,	90.168	1	.10	1
21	-46.806,	88.866	1	.10	1
22	-57.851,	83.363	1	.10	1
23	43.318,	80.772	1	.10	1
24	48.826,	91.667	1	.10	1
25	-71.320,	98.561	1	.10	1
26	78.872,	99.119	1	.10	1
27	20.154,	88.391	1	.10	1
28	70.283,	94.964	1	.10	1
29	40.841,	80.341	1	.10	1
30	57.020,	94.491	1	.10	1
31	40.921,	93.282	1	.10	1
32	70.329,	89.944	1	.10	1
33	-82.701,	101.149	1	.10	1
34	33.316,	99.722	1	.10	1
35	36.723,	84.105	1	.10	1
36	6.294,	78.877	1	.10	1
37	77.223,	93.198	1	.10	1
38	-64.730,	88.443	1	.10	1
39	-16.030,	92.074	1	.10	1
40	-58.878,	84.873	1	.10	1
41	-36.899,	82.567	1	.10	1
42	13.860,	93.823	1	.10	1
43	12.322,	86.024	1	.10	1
44	-80.383,	79.724	1	.10	1
45	-2.634,	77.549	1	.10	1
46	78.536,	97.466	1	.10	1
47	-80.090,	81.977	1	.10	1
48	40.528,	81.637	1	.10	1
49	-73.864,	81.715	1	.10	1
50	-69.100,	81.396	1	.10	2

num	x	y	status	radius	type
51	75.536,	82.485	1	.10	2
52	70.348,	86.353	1	.10	2
53	-80.392,	99.217	1	.10	2
54	31.496,	80.909	1	.10	2
55	-80.649,	89.850	1	.10	2
56	84.710,	84.882	1	.10	2
57	-12.054,	86.260	1	.10	2
58	33.566,	95.306	1	.10	2
59	82.227,	83.276	1	.10	2
60	-27.088,	81.566	1	.10	2
61	-79.012,	101.078	1	.10	2
62	-7.314,	81.453	1	.10	2
63	-56.933,	93.084	1	.10	2
64	39.837,	78.599	1	.10	2
65	-49.200,	90.353	1	.10	2
66	-31.002,	79.554	1	.10	2
67	6.958,	83.291	1	.10	2
68	42.579,	87.717	1	.10	2
69	2.330,	98.190	0	.10	2
70	48.239,	77.894	1	.10	2
71	44.045,	100.627	1	.10	2
72	-32.608,	96.975	1	.10	2
73	6.853,	85.755	1	.10	2
74	-70.743,	98.157	1	.10	2
75	89.132,	90.555	1	.10	2
76	-82.859,	100.557	1	.10	2
77	52.129,	96.808	1	.10	2
78	-2.663,	97.014	1	.10	2
79	26.994,	84.669	1	.10	2
80	62.457,	97.097	1	.10	3
81	-37.272,	86.905	1	.10	3
82	4.328,	93.402	1	.10	3
83	-23.196,	78.691	1	.10	3
84	5.974,	100.643	1	.10	3
85	77.088,	99.466	1	.10	3
86	81.930,	101.050	1	.10	3
87	-61.883,	80.105	1	.10	3
88	-47.431,	101.284	1	.10	3

Mine vs Vehicle Pk Values

Veh Type\	Mine Type 1	2	3	4	5
1	.05	.05	.50	0.	0.
2	.10	.50	.70	0.	0.
3	.90	.90	.90	0.	0.
4	0.	0.	0.	0.	0.
5	0.	0.	0.	0.	0.

DUMP OF VEHICLE DATA

ELEMENT STATUS AT
TIME = 2.831

ID	X	Y	SPEED	STATUS	TYPE
1	0.	160.00	0.	1	1
2	-.27	160.00	0.	1	3
3	-.20	160.00	0.	1	3
4	.49	98.17	0.	0	3
5	-.94	160.00	0.	1	3
6	-.41	160.00	0.	1	3

MOVEMENT PLANS

ELEMENT 1

X-CORR	Y-CORR	SLOPE	INTERCEPT
0.	0.	+1.E+006	0.
0.	20.00	+1.E+006	20.000
0.	40.00	+1.E+006	40.000
0.	60.00	+1.E+006	60.000
0.	80.00	+1.E+006	80.000
0.	100.00	+1.E+006	100.000
0.	120.00	+1.E+006	120.000
0.	140.00	+1.E+006	140.000
0.	160.00	0.	0.

ELEMENT 2

X-CORR	Y-CORR	SLOPE	INTERCEPT
-.064	0.	-36.041	-2.302
-.619	20.00	17.002	30.520
.558	40.00	-59.507	73.179
.221	60.00	-44.127	69.773
-.232	80.00	29.323	86.796
.450	100.00	-52.801	123.776
.072	120.00	110.978	112.063
.252	140.00	-38.433	149.675
-.269	160.00	0.	0.

ELEMENT 3

X-CORR	Y-CORR	SLOPE	INTERCEPT
-1.265	0.	48.836	61.772
-.855	20.00	25.819	42.085
-.081	40.00	-53.199	35.705
-.457	60.00	36.019	76.449
.099	80.00	-148.843	94.673
-.036	100.00	36.606	101.310
.511	120.00	953.300	-366.723
.532	140.00	-27.263	154.491
-.202	160.00	0.	0.

ELEMENT 4

X-CORR	Y-CORR	SLOPE	INTERCEPT
.228,	0. ,	-15.653,	3.567
-1.050,	20.00,	26.428,	47.744
-.293,	40.00,	-2.E+003,	-464.771
-.305,	60.00,	20.086,	66.119
.691,	80.00,	-92.499,	143.925
.475,	100.00,	-26.237,	112.459
-.287,	120.00,	-42.199,	107.871
-.761,	140.00,	34.208,	166.045
-.177,	160.00,	0. ,	0.

ELEMENT 5

X-CORR	Y-CORR	SLOPE	INTERCEPT
-.209,	0. ,	57.200,	11.934
.141,	20.00,	62.933,	11.126
.459,	40.00,	-35.104,	56.106
-.111,	60.00,	98.160,	70.888
.093,	80.00,	-226.818,	101.055
.005,	100.00,	670.616,	96.882
.034,	120.00,	-45.640,	121.573
-.404,	140.00,	-37.392,	124.903
-.939,	160.00,	0. ,	0.

ELEMENT 6

X-CORR	Y-CORR	SLOPE	INTERCEPT
.206,	0. ,	19.616,	-4.038
1.225,	20.00,	-24.161,	49.607
.398,	40.00,	-291.532,	155.917
.329,	60.00,	-81.747,	86.896
.084,	80.00,	31.156,	77.372
.726,	100.00,	-87.805,	163.772
.499,	120.00,	-22.568,	131.251
-.388,	140.00,	-900.910,	-209.266
-.410,	160.00,	0. ,	0.

ELEMENT STATUS AT END OF RUN

NUM	STATUS	X	Y
1	ALIVE	0.	160.000
2	ALIVE	-.269	160.000
3	ALIVE	-.202	160.000
4	DEAD	.495	98.170
5	ALIVE	-.939	160.000
6	ALIVE	-.410	160.000

OBSTACLE LOCATIONS

NUM	X	Y	RADIUS
4	.4947	98.1701	3.16

BYPASS MAPS

ELE	E-X	E-Y	ST	P1-X	P1-Y	ST	P2-X	P2-Y	ST	Ent-X	Ent.Y	ST
1	0.	0.	0	0.	0.	0	0.	0.	0	0.	0.	0
2	0.	0.	0	0.	0.	0	0.	0.	0	0.	0.	0
3	0.	0.	0	0.	0.	0	0.	0.	0	0.	0.	0
4	0.	0.	0	0.	0.	0	0.	0.	0	0.	0.	0
5	.04	91.13	0	-5.83	94.52	0	-5.83	101.82	1	.01	105.75	0
6	.43	90.90	0	-5.83	94.52	0	-5.83	101.82	1	.67	105.38	0

```
FULL WIDTH PLOWS moved   4.00 mines.

MINE EFFECTS
Type 1 mines had   0.   kills (pressure mines)
Type 2 mines had   1.00 kills (influence mines)
Type 3 mines had   0.   kills (contact mines)
Type 4 mines had   0.   kills
Type 5 mines had   0.   kills

Total mine kills were    1.00


      SUMMARY DATA
     TASKFORCE SIZE
     --------- ----
Total number of elements was    6
Number of Full Plows =   1
Number of Track Plows =   0
Number of Track Type 1 =   5
Number of Track Type 2 =   0
Number of Track Type 3 =   0

     Taskforce Velocity
Full plow has a velocity of   6.000 kph
Track type 1 has a velocity of   8.000 kph

       MINE DATA
Number of Pressure Mines =   50
Number of Influence Mines =   30
Number of Contact Mines =    8
Number of Type 4 Mines =    0
Number of Type 5 Mines =    0


             MINE DENSITIES
AREA DENSITY
   TYPE 1 MINES =    .011
   TYPE 2 MINES =    .007
   TYPE 3 MINES =    .002
   TYPE 4 MINES =   0.
   TYPE 5 MINES =   0.

TOTAL ALL MINES =    .020


LINEAR DENSITY
   TYPE 1 MINES =    .278
   TYPE 2 MINES =    .167
   TYPE 3 MINES =    .044
   TYPE 4 MINES =   0.
   TYPE 5 MINES =   0.

TOTAL ALL MINES =    .489

           Mine vs Vehicle Pk Values
                 Mine Type
Veh Type\  1      2      3      4      5
-----------------------------------------
    1     .05    .05    .50   0.     0.
    2     .10    .50    .70   0.     0.
    3     .90    .90    .90   0.     0.
    4    0.     0.     0.     0.     0.
    5    0.     0.     0.     0.     0.
```

The simulation made 1 runs.

The full width plow used had a width of 4.50 and an effectiveness of 1.00

SUMMARY STATISTICS

MINES MOVED BY PLOW

An average of 2.50 mines were displaced by the plow each run

MINE KILLS BY MINE TYPE

Mean mine kills by type 1 mines was 0. , with variance 0.
Mean mine kills by type 2 mines was 1.00, with variance 0.
Mean mine kills by type 3 mines was 0. , with variance 0.
Mean mine kills by type 4 mines was 0. , with variance 0.
Mean mine kills by type 5 mines was 0. , with variance 0.

Mean mine kills for all mine types 1.00, with variance 0.

DIRECT FIRE STATISTICS

RED OVERWATCH

TYPE	STATUS	FIRE RATE	MEAN ACQ RATE
1	1	.500	.700

Red fired 3.0 direct fire shots.
accomplishing 0. kills of blue taskforce vehicles.

Red direct fire overwatch killed a mean of 0. with a variance of 0. taskforce vehicles per run.

TOTAL TASKFORCE LOSSES

The mean taskforce loss rate was .17 with a variance of .00

APPENDIX F - EXAMPLE MULTIPLE RUN RESULTS

The model collects designated data throughout the course of a data run. Following is an example of one such run.
The raw data columns consist of the following data elements.

A = Iteration number
B = Mines Kills
C = Direct Fire Kills
D = Total Kills
E = Mine Kills while bypassing Obstruction
F = Time of Death for Vehicle 1 (0 if survives breach)
G = Time of Death for Vehicle 2 (0 if survives breach)
H = Time of Death for Vehicle 3 (0 if survives breach)
I = Time of Death for Vehicle 4 (0 if survives breach)
J = Time of Death for Vehicle 5 (0 if survives breach)
K = Time of Death for Vehicle 6 (0 if survives breach)
L = Time of Completion for Vehicle 1
M = Time of Completion for last surviving vehicle

RAW DATA

A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
2	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
3	3	0	3	2	0.	0.	0.	3.53	3.80	3.98	1.60	3.90
4	1	0	1	0	0.	3.17	0.	0.	0.	0.	1.60	4.62
5	0	1	1	0	1.57	0.	0.	0.	0.	0.	0.	4.72
6	3	1	4	1	1.58	3.19	3.39	0.	0.	3.99	0.	4.48
7	5	1	6	5	.81	3.14	3.38	3.67	3.80	3.99	0.	0.
8	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
9	1	0	1	0	0.	0.	3.31	0.	0.	0.	1.60	4.62
10	1	0	1	0	0.	0.	0.	3.54	0.	0.	1.60	4.62
11	1	0	1	0	0.	0.	0.	0.	3.80	0.	1.60	4.62
12	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
13	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
14	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
15	2	1	3	2	.82	3.16	3.43	0.	0.	0.	0.	4.59
16	4	1	5	3	.83	3.19	3.38	3.58	3.81	0.	0.	4.67
17	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
18	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
19	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
20	4	0	4	3	0.	0.	3.32	3.56	3.84	4.01	1.60	3.68
21	2	0	2	0	0.	3.18	0.	0.	0.	3.97	1.60	4.32
22	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
23	1	2	3	0	1.27	3.06	0.	0.	0.	3.98	0.	4.44
24	4	0	4	3	0.	0.	3.33	3.64	3.89	4.05	1.60	3.68
25	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
26	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
27	1	0	1	0	0.	3.19	0.	0.	0.	0.	1.60	4.62
28	3	0	3	1	0.	0.	3.30	3.55	0.	4.18	1.60	4.41
29	3	0	3	2	0.	0.	3.31	3.54	3.82	0.	1.60	4.67

30	3	0	3	2	0.	3.23	3.44	0.	3.84	0.	1.60	4.57
31	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
32	1	0	1	0	0.	0.	3.46	0.	0.	0.	1.60	4.56
33	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
34	4	1	5	3	1.12	0.	3.32	3.60	3.88	4.16	0.	3.73
35	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
36	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
37	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
38	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
39	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
40	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
41	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
42	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
43	1	0	1	0	0.	3.12	0.	0.	0.	0.	1.60	4.63
44	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
45	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
46	1	0	1	0	0.	0.	0.	0.	0.	3.97	1.60	4.35
47	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
48	1	0	1	0	0.	0.	0.	3.64	0.	0.	1.60	4.62
49	1	0	1	0	0.	0.	0.	3.56	0.	0.	1.60	4.62
50	1	0	1	0	0.	0.	0.	3.69	0.	0.	1.60	4.62
51	2	1	3	1	1.15	3.14	3.41	0.	0.	0.	0.	4.61
52	4	0	4	3	0.	0.	3.35	3.56	3.90	4.03	1.60	3.68
53	4	1	5	4	1.18	0.	3.43	3.65	3.92	4.07	0.	3.73
54	0	1	1	0	1.50	0.	0.	0.	0.	0.	0.	4.72
55	1	0	1	0	0.	0.	0.	0.	0.	3.96	1.60	4.35
56	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
57	3	1	4	2	.76	3.14	0.	0.	3.80	4.04	0.	4.22
58	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
59	1	0	1	0	0.	3.08	0.	0.	0.	0.	1.60	4.62
60	3	1	4	3	0.	3.27	3.49	3.75	3.89	0.	1.60	4.65
61	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
62	4	2	6	2	.74	2.75	3.48	3.65	3.95	4.14	0.	0.
63	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
64	2	0	2	1	.98	3.26	0.	0.	0.	0.	0.	4.62
65	4	0	4	2	0.	0.	3.45	3.57	3.80	4.08	1.60	3.68
66	1	0	1	0	0.	0.	0.	3.65	0.	0.	1.60	4.62
67	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
68	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
69	1	0	1	0	0.	0.	0.	3.55	0.	0.	1.60	4.62
70	3	0	3	2	0.	0.	0.	3.63	3.91	4.07	1.60	3.90
71	5	0	5	4	0.	3.16	3.35	3.60	3.82	4.02	1.60	1.60
72	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
73	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
74	1	0	1	0	0.	0.	0.	0.	0.	3.99	1.60	4.35
75	1	0	1	0	0.	3.06	0.	0.	0.	0.	1.60	4.62
76	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
77	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
78	2	1	3	1	.76	3.21	0.	0.	0.	4.10	0.	4.44
79	2	0	2	1	0.	0.	0.	0.	3.75	3.99	1.60	4.13
80	1	0	1	0	0.	0.	3.39	0.	0.	0.	1.60	4.55
81	2	0	2	1	0.	3.10	3.42	0.	0.	0.	1.60	4.63
82	2	1	3	1	.86	3.26	3.49	0.	0.	0.	0.	4.59
83	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
84	5	0	5	4	0.	3.23	3.44	3.67	3.86	4.11	1.60	1.60
85	1	1	2	0	0.	3.34	0.	0.	0.	3.96	1.60	4.44
86	0	1	1	0	1.28	0.	0.	0.	0.	0.	0.	4.63
87	2	1	3	2	1.11	3.21	3.48	0.	0.	0.	0.	4.64
88	1	1	2	1	1.03	3.22	0.	0.	0.	0.	0.	4.62
89	3	0	3	2	0.	0.	3.28	3.56	3.86	0.	1.60	4.69
90	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
91	1	0	1	0	0.	3.23	0.	0.	0.	0.	1.60	4.62
92	3	1	4	2	.75	0.	0.	3.66	3.97	4.11	0.	3.96
93	1	0	1	0	0.	0.	0.	0.	3.89	0.	1.60	4.62
94	3	0	3	2	0.	0.	0.	3.53	3.75	4.00	1.60	3.90
95	3	1	4	2	1.36	0.	3.35	0.	3.90	4.04	0.	4.24
96	4	1	5	4	1.13	0.	3.44	3.62	3.83	3.95	0.	3.72

97	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
98	3	0	3	1	0.	3.09	3.46	3.59	0.	0.	1.60	4.67
99	1	1	2	0	.78	3.17	0.	0.	0.	0.	0.	4.67
100	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
101	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
102	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
103	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
104	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
105	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
106	3	0	3	1	1.02	0.	0.	0.	3.76	4.01	0.	4.17
107	3	1	4	3	.80	3.15	3.31	3.60	0.	0.	0.	4.69
108	1	0	1	0	0.	0.	0.	0.	0.	3.98	1.60	4.35
109	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
110	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
111	2	0	2	1	0.	3.18	0.	3.67	0.	0.	1.60	4.64
112	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
113	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
114	3	0	3	2	0.	0.	0.	3.58	3.85	4.08	1.60	3.90
115	4	0	4	2	0.	3.21	0.	3.51	3.70	4.13	1.60	3.95
116	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
117	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58
118	2	0	2	0	0.	3.06	3.45	0.	0.	0.	1.60	4.63
119	2	1	3	2	1.19	3.23	3.48	0.	0.	0.	0.	4.65
120	0	0	0	0	0.	0.	0.	0.	0.	0.	1.60	4.58

SUMMARY DATA

TASKFORCE SIZE

Total number of elements was 6
Number of Full Plows = 1
Number of Track Plows = 0
Number of Track Type 1 = 5
Number of Track Type 2 = 0
Number of Track Type 3 = 0

Taskforce Velocity
Full plow has a velocity of 6.000 kph
Track type 1 has a velocity of 8.000 kph

MINE DATA
Number of Pressure Mines = 50
Number of Influence Mines = 30
Number of Contact Mines = 8
Number of Type 4 Mines = 0
Number of Type 5 Mines = 0

MINE DENSITIES
AREA DENSITY
TYPE 1 MINES = .011
TYPE 2 MINES = .007
TYPE 3 MINES = .002
TYPE 4 MINES = 0.
TYPE 5 MINES = 0.
TOTAL ALL MINES = .020

LINEAR DENSITY
TYPE 1 MINES = .278
TYPE 2 MINES = .167
TYPE 3 MINES = .044
TYPE 4 MINES = 0.
TYPE 5 MINES = 0.
TOTAL ALL MINES = .489

Mine vs Vehicle Pk Values

	Mine Type				
Veh Type\	1	2	3	4	5
1	.05	.05	.50	0.	0.
2	.10	.50	.70	0.	0.
3	.90	.90	.90	0.	0.
4	0.	0.	0.	0.	0.
5	0.	0.	0.	0.	0.

RED OVERWATCH

TYPE	STATUS	FIRE RATE	MEAN ACQ RATE
1	1	.500	.700

The simulation made 120 runs.
The full width plow used had a width of 4.50 and an effectiveness of 1.00

SUMMARY STATISTICS

MINES MOVED BY PLOW
An average of 1.18 mines were displaced by the plow each run

MINE KILLS BY MINE TYPE

Mean mine kills by type 1 mines was	.70,	with variance	.90
Mean mine kills by type 2 mines was	.29,	with variance	.28
Mean mine kills by type 3 mines was	.19,	with variance	.18
Mean mine kills by type 4 mines was	0. ,	with variance	0.
Mean mine kills by type 5 mines was	0. ,	with variance	0.
Mean mine kills for all mine types	1.06,	with variance	1.52

DIRECT FIRE STATS

Red direct fire overwatch fired a mean of 1.19 with a variance of 1.17 shots per run.

Red direct fire overwatch killed a mean of .20 with a variance of .19 taskforce vehicles per run.

The mean taskforce loss rate was 1.26 with a variance of .60

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center ATTN: Reference Services Branch (DTIC-DFRA) Building 5, Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Deputy Undersecretary of the Army for Operations Research Room 2E261, The Pentagon Washington, D.C. 20310	1
4. Professor Samuel H. Parry Department of Operations Research Naval Postgraduate School, Code OR/Py Monterey, CA 93943-5000	2
5. LTC William Caldwell Department of Operations Research Naval Postgraduate School, Code OR/CW Monterey, CA 93943-5000	1
6. Commander and Director U.S. Army Engineers Waterways Experiment Station ATTN: CEWES-GM-L (Dr. John Farr) 3909 Halls Ferry Road Vicksburg, MS 39180-6199	1
7. Director Methodology and Analysis Directorate US Army Testing and Experimentation Command Ft. Hood, TX 76544	1
8. Director U.S. Army TRADOC Analysis Command - Ft. Leavenworth Attn: ATRC-FOQ (Technical Information Center) Fort Leavenworth, KS 66027-5200	1

9. Commandant
U.S. Army Engineer School
ATTN: ATSE-CDC-M (MAJ Dave Davis)
Fort Leonard Wood, MO 65473-6600 — 1

10. Commandant
U.S. Army Armor School
ATTN: ATZK-AR-P
Fort Knox, KY 40121-5187 — 1

11. MAJ Alan Anderson
508 Camelia Lane
DeLand, FL 32724 — 2

Thesis
A44444
c.1

Anderson
Methodologies for the high resolution modeling of minefield dynamics.

Thesis
A44444
c.1

Anderson
Methodologies for the high resolution modeling of minefield dynamics.

3 2768 00031888 5